```
else
        State ← Next_State_LPS(State).
```

In the case, where the current state corresponds to a probability value of 0.5, which corresponds to the *State* index of 0, and a LPS symbol is observed, the sense of MPS and LPS has to be interchanged.

Table 9-35 – Probability transition

State	Next_State_MPS_INTRA	Next_State_MPS	Next_State_LPS	State	Next_State_MPS	Next_State_LPS
0	1	1	0	32	33	22
1	2	2	0	33	34	22
2	3	3	1	34	35	23
3	4	4	2	35	36	23
4	5	5	2	36	37	24
5	6	6	3	37	38	24
6	7	7	4	38	39	25
7	8	8	5	39	40	25
8	9	9	6	40	41	26
9	10	10	7	41	42	26
10	11	11	8	42	43	27
11	12	12	8	43	44	27
12	13	13	10	44	45	28
13	14	14	10	45	46	28
14	15	15	10	46	47	29
15	16	16	11	47	48	29
16	17	17	12	48	49	30
17	18	18	13	49	50	30
18	19	19	14	50	51	30
19	20	20	14	51	52	31
20	21	21	14	52	53	32
21	22	22	14	53	54	33
22	23	23	15	54	55	33
23	23	24	16	55	56	34
24	-/-	25	17	56	57	34
25	-/-	26	18	57	58	35
26	-/-	27	19	58	59	35
27	-/-	28	19	59	60	36
28	-/-	29	20	60	61	37
29	-/-	30	20	61	62	37
30	-/-	31	21	62	63	38

31	-/-	32	21	63	63	38

Table 9-36 – RTAB[State][Q] table for interval subdivision

State	0	1	2	3	State	0	1	2	3
0	9216	11264	13312	15360	32	896	1152	1344	1536
1	8832	10816	12800	14720	33	896	1088	1280	1472
2	8512	10368	12288	14144	34	832	1024	1216	1408
3	8128	9920	11712	13504	35	832	960	1152	1344
4	7680	9344	11072	12736	36	768	960	1088	1280
5	7168	8768	10368	11968	37	768	896	1088	1216
6	6912	8448	9984	11520	38	704	896	1024	1152
7	6336	7808	9216	10624	39	704	832	960	1152
8	5888	7232	8512	9856	40	640	832	960	1088
9	5440	6656	7872	9088	41	640	768	896	1088
10	5120	6208	7360	8512	42	640	768	896	1024
11	4608	5632	6656	7680	43	576	704	832	960
12	4224	5184	6144	7104	44	576	704	832	960
13	3968	4800	5696	6592	45	576	704	832	960
14	3712	4480	5312	6144	46	512	640	768	896
15	3456	4224	4992	5760	47	512	640	768	896
16	3072	3776	4416	5120	48	512	640	768	832
17	2816	3456	4096	4736	49	512	640	704	832
18	2624	3200	3776	4416	50	512	576	704	832
19	2432	3008	3520	4096	51	448	576	704	768
20	2304	2816	3328	3840	52	448	576	640	768
21	2048	2496	2944	3392	53	448	512	640	704
22	1856	2240	2688	3072	54	448	512	640	704
23	1664	2048	2432	2816	55	448	512	576	704
24	1536	1856	2240	2560	56	384	512	576	704
25	1408	1728	2048	2368	57	384	512	576	640
26	1344	1600	1920	2176	58	384	448	576	640
27	1216	1472	1792	2048	59	384	448	576	640
28	1152	1408	1664	1920	60	384	448	512	640
29	1088	1344	1536	1792	61	384	448	512	640
30	1024	1280	1472	1728	62	384	448	512	576
31	960	1216	1408	1600	63	384	448	512	576

9.2.4.3 Description of the arithmetic decoding engine

The status of the arithmetic decoding engine is represented by a value V pointing into the code sub-interval and the corresponding range R of that sub-interval. Figure 9-3 gives an illustration of the whole decoding process. Performing the InitDecoder procedure, which is further specified in subclause 9.2.4.3.1, appropriately initialises V and R. For decoding of each single decision S, the following two-step operation is employed: First, the related context model CTX is determined according to the rules specified in subclauses 9.2.2. Given the context model CTX, the decoding operation Decode(CTX) then delivers the decoded symbol S as is described in detail in subclause 9.2.4.3.2.

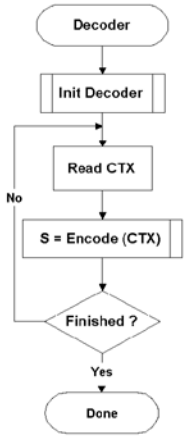

Figure 9-3 - Overview of the Decoding Process

9.2.4.3.1 Initialisation of the decoding engine

In the initialisation procedure of the decoder, as illustrated in Figure 9-4, V is first filled with two bytes of the compressed data using the GetByte routine as specified in subclause 9.2.4.3.4, and then the range R is set to 0x8000.

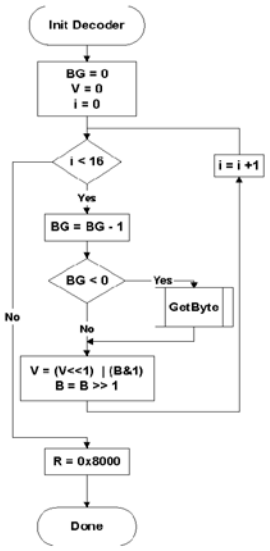

Figure 9-4 – Flowchart of initialisation of the decoding engine

9.2.4.3.2 Decoding a decision

Figure 9-5 shows the flowchart for decoding a single decision. In a first step, the estimation of the sub-interval ranges R_{LPS} and R_{MPS} corresponding to the LPS and the MPS decision is performed as follows.

Given the interval range R, we first map R to a quantized value Q using

$$Q=(R-0x4001)>>12, \tag{9-11}$$

such that the state index *State* and Q are used as an entry in the look-up table RTAB to determine R_{LPS}:

$$R_{LPS}=RTAB[State][Q]. \tag{9-12}$$

Table 9-36 specifies the corresponding values of RTAB in 16-bit representation. The tabulated values are actually given in 8-bit accuracy; the maximum value of RTAB corresponds to 14 bits and all values have been left-shifted by 6 bits for a better access in a 16-bit architecture.

In a second step, the current value of V is compared to the size of the MPS sub-interval R_{MPS}. If V is greater than or equal to R_{MPS} a LPS is decoded, V is decremented by R_{MPS} and the new range R is set to R_{LPS}; otherwise a MPS is decoded and the new range R is determined to be R_{MPS}. Given the decoded decision, the probability update is performed accordingly as specified in subclause 9.2.4.2. Depending on the current value of the new range R, renormalization will be performed as described in more detail in subclause 9.2.4.3.3.

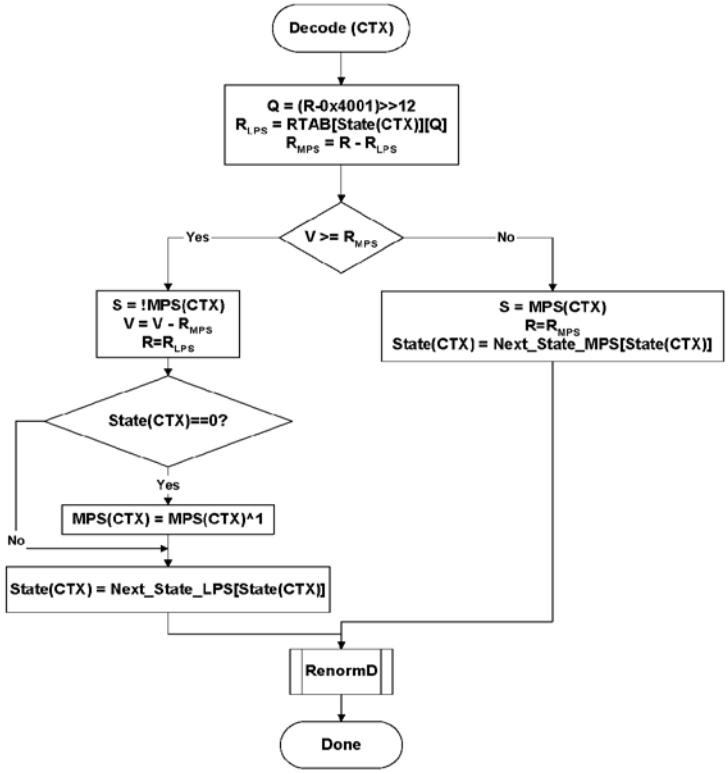

Figure 9-5 – Flowchart for decoding a decision

9.2.4.3.3 Renormalization in the decoding engine (RenormD)

Renormalization is illustrated in Figure 9-6. The current range R is first logically compared to 0x4000: If it is greater than that value, no renormalization is needed and RenormD is finished; otherwise the renormalization loop is entered. Within this loop, the range R is doubled, i.e. left-shifted by 1 and the bit-counter BG is decremented by 1. In the case, that the condition $BG<0$ holds, a new byte of compressed is inserted into B by calling the GetByte routine. Finally, the next bit of B is shifted into V.

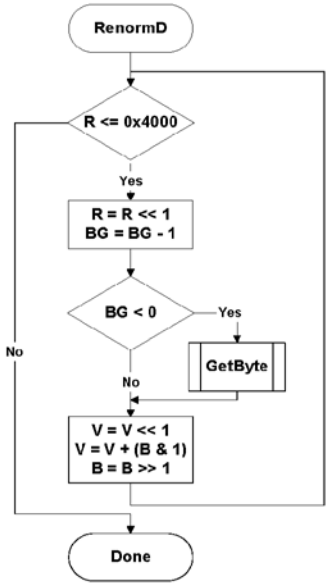

Figure 9-6 – Flowchart of renormalization

9.2.4.3.4 Input of compressed bytes (GetByte)

Figure 9-7 shows how the input of compressed data is performed. At the initialsation stage of the whole decoding process or in case a renormalization occurs and the bit-counter *BG* has a negative value, this procedure will be invoked. First, a new byte of compressed data is read out of the bitstream *C*; then the index *CL* pointing to the current position of the bitstream *C* is incremented by 1 and the bit-counter is set to 7.

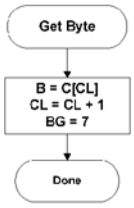

Figure 9-7 – Flowchart for Input of Compressed Bytes

9.2.4.3.5 Decoder bypass for decisions with uniform pdf (Decode_eq_prob)

This special decoding routine applies to decoding of the sign information of motion vector data and the sign of the levels of significant transform coefficients, which are assumed to have a uniform probability distribution. Consequently omitting the probability estimation in this special case reduces the decoding process to a single comparison ($V>=R_{half}$?) in order to determine the right subinterval and its corresponding decoded symbol value *S*. The subsequent renormalization process is similar to that performed in the renormalization procedure RenormD, as depicted in Figure 9-8 with two modifications. Firstly, the rescaling operation $R \leftarrow (R<<1)$ is unnecessary and secondly, the initial comparison ($R<=0x4000$?) can be omitted.

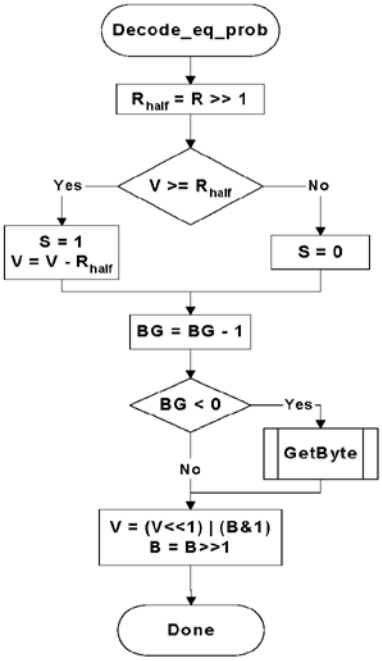

Figure 9-8 – Flowchart of decoding bypass

10 Decoding process for B slices

10.1 Introduction

The use of B (bi-predictive) slices is indicated in the nal_unit_type. A B slice is an inter predicted slice. The major difference between a B slice and P slice is that B slices are coded in a manner in which some macroblocks or blocks may use a weighted average of two distinct inter prediction values for building the prediction signal. Generally, B slices utilize two distinct reference index lists. Each of these index lists refer to pictures in the reference picture buffer.

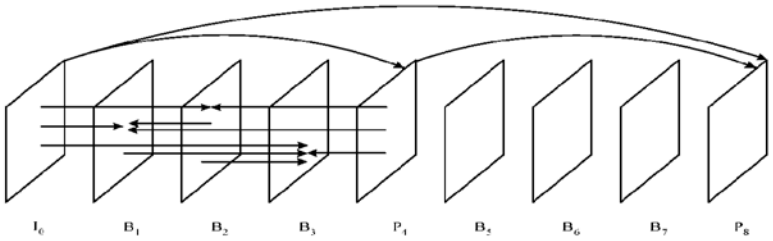

Figure 10-1 – Illustration of B picture concept

NOTE - The location of pictures in the bitstream is in a decoding order. Pictures that are dependent on other pictures shall occur in the bitstream after the pictures on which they depend. Figure 10-1 shows one hypothetical example, where three B pictures are inserted in output order between an I and a P picture and the subscripts indicate the output order. The P picture P_4 only depends on the first Intra picture I_0. The B picture B_2, which is temporally located between I_0 and P_4, depends on both of these pictures. The B picture B_1 depends on I_0, P_4, and B_2; the B picture B_3 additionally depends on B_1. While the output order for this example is given by I_0, B_1, B_2, B_3, P_4, the decoding order is I_0, P_4, B_2, B_1, B_3.

10.2 Decoding process for macroblock types and sub macroblock types

There are five different prediction modes supported by B pictures. They are the list 0, list 1, bi-predictive, direct, and intra prediction modes. In bi-predictive mode, the prediction signal is formed by a weighted average of list 0 and list 1 prediction values. The direct mode can result in prediction modes list 0, list 1, or bi-predictive. Prediction using the direct mode is derived from a combination of the motion vectors and macroblock type used in the co-located macroblock of the first picture (the picture at index 0) in list 1.

Coded macroblocks in B pictures utilize a similar tree-structured macroblock partitioning to P pictures. Depending on the number of elements in the two reference picture lists, up to two reference picture indices are coded for each bi-predicted region. Additionally, for each luma block of 16x16, 16x8, 8x16 samples and the associated chroma blocks, and each sub macroblock, the prediction mode (list 0, list 1, bi-predictive, direct, intra) can be chosen separately. In order to avoid a separate codeword to specify the prediction mode,the indication of the prediction direction is incorporated into the codewords for macroblock type and sub macroblock type, respectively, as shown in the Table 7-12 and Table 7-17. A sub macroblock of a B picture macroblock can also be coded in direct mode.

When mb_adaptive_frame_field_flag = = 1, the current direct-mode macroblock shall follow the same frame/field coding mode as its co-located macroblock.

10.3 Decoding process for motion vectors

10.3.1 Differential motion vectors

In the following, the terms *temporal ordering* and *temporal distance* refer to ordering and distance according to the picture order counter described in subclause 8.3.2. This is also the decoder output order and the intended display order.

Motion vectors for list 0, list 1, or bi-predictive blocks are differentially encoded. A prediction has to be added to the motion vector differences in order to reconstruct motion vectors for the current macroblock.

As a special case of bi-predictive blocks, if the two reference pictures used for the bi-prediction both occur temporally earlier or both occur temporally later than the current picture being decoded, then the motion vector decoding is performed as described in subclause 10.3.2.

Otherwise, the predictions for bi-predictive blocks are formed from the motion vectors of spatially neighbouring blocks in a way similar to that described in subclause 8.4.1, but with a few important distinctions.

First, a list i, where i = 0 or 1, motion vector MV_i from the current block is predicted only from neighbouring blocks that contain motion vectors with the same temporal direction (earlier or later in time) as MV_i. If a neighbouring block does not have a motion vector with the same temporal direction, the predictor for that block is set to zero and the neighbouring block shall be considered as belonging to a *different reference picture* for purposes of computing the median prediction of subclause 8.4.1.1.

Second, if a neighbouring bi-predicted block has both motion vectors pointing in the same temporal direction as MV_i, and both motion vectors point to the same reference picture, then the list 0 motion vector from that block is used as a prediction. Otherwise, if a neighbouring bi-predicted block has both motion vectors pointing in the same temporal

direction as MV_i but they point to different reference pictures, then the motion vector that points to the temporally closest reference picture is used.

Third, reconstructed motion vectors in direct mode neighbouring blocks shall be used as predictions for the current block motion vectors.

If a direct mode neighbouring block has two motion vectors, then this block is treated as if it were a bi-predictive neighbouring block.

If a direct mode neighbouring block has only one motion vector, then this block is considered as a list 0 or list 1 block.

In the case that the co-located block in such a direct mode block is intra coded and the direct_spatial_mv_pred_flag is 0, the direct mode block is treated as belonging to a *different reference picture* for purposes of computing the median prediction of subclause 8.4.1.1.

10.3.2 Motion vector decoding with scaled MV

If both reference pictures (ref_idx_l0 & ref_idx_l1) occur temporally earlier or both occur temporally later than the current picture being decoded, then the following process is followed for decoding the motion vectors MV_1 and MV_2 for bi-predictive modes. The motion vector decoding process is illustrated in Figure 10-2. The motion vector MV_1 for the first reference picture (ref_idx_l0) is differentially decoded using motion vector prediction as described in ~~8.4.1~~10.3.1. However, the method used for decoding the motion vector MV_2 for the second reference picture (ref_idx_l1) is as follows:

The scaled motion vector (*smv*) is first calculated from the motion vector MV_1 as:

$$Z = (TD_2 \times 256) / TD_1 \qquad smv = (Z \times MV_1 + 128) >> 8$$

where TD_1 is the temporal distance between the current picture and the reference picture indicated by ref_idx_l0, and TD_2 is the temporal distance between the current picture and the reference picture indicated by ref_idx_l1.

Then, MV_2 is differentially coded with respect to *smv*.

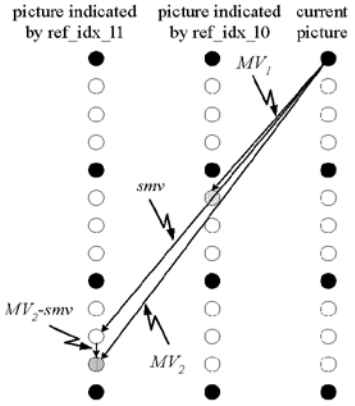

Figure 10-2 – Differential motion vector decoding with scaled motion vector

10.3.3 Motion vectors in direct mode

The direct_spatial_mv_pred_flag identifies for the current slice whether the direct mode motion vectors are calculated using a spatial or temporal technique. If this indicator is set to 1 then the spatial technique is used. Otherwise, if this indicator is set to 0, then direct mode motion vectors are calculated using the temporal technique.

10.3.3.1 Spatial technique of obtaining the direct mode motion parameters

The first step in the spatial technique for direct mode prediction is the determination of a candidate reference picture index for each list (list 0 and list 1).

The reference picture indices for the neighbouring blocks A, B, C, and D within the current slice for the 16x16 current luma block E, as described in subclause 8.4.1.1 and shown in Figure 8-4, shall be used to determine a preliminary candidate reference picture index for each list. The preliminary candidate reference picture index for each list (list 0 and list 1) shall be the minimum reference picture index among the reference picture indices used from the same list for the prediction or bi-prediction of the neighbouring blocks. If no neighbouring blocks are present within the current slice that use prediction from the same list, either for prediction or bi-prediction, the preliminary candidate reference index for that list shall be interpreted as not existing.

If a preliminary candidate reference index exists for either list, decision of the final candidate reference indices and associated motion vector values for that list for each 4x4 block of the current macroblock depends on the coded parameters of the co-located 4x4 blocks in the first picture in list 1. If the first picture in list 1 is a short-term picture and if all lines of the co-located 4x4 block were predicted using list 0 prediction with reference picture index 0 and motion vector components in the range of -1 to 1 inclusive, then the final candidate reference picture index for the list of the current 4x4 block shall be 0 and shall be associated with a candidate motion vector value of (0, 0). Otherwise, the final candidate reference picture index for the list shall be the preliminary candidate reference picture index for the list and the associated candidate motion vector shall be obtained, using the 16x16 block motion vector prediction, as described in subclause 8.4.1 by using the final reference picture index.

If both candidate reference picture indices exist, then the block is predicted as a bi-prediction block using the final candidate reference picture index and motion vector for each list. Otherwise, if a candidate reference picture index exists for only one of the two lists, the block shall be predicted by single-list prediction using the final candidate reference picture index and associated motion vector for the existing candidate. Finally, if neither candidate reference picture index exists, bi-prediction shall be used with reference picture index zero and associated motion vector (0, 0) for both lists.

10.3.3.2 Temporal technique of obtaining the direct mode motion parameters

In the temporal technique direct mode, the same block structure as for the co-located macroblock of the first picture (the picture at index 0) in list 1 is used. For each block of the current macroblock, the list 0 and list 1 motion vectors are computed as scaled versions of the list 0 motion vector of the co-located block in the list 1 reference picture as described below.

The list 0 reference picture for the direct mode current block is the same as the list 0 reference picture used for the co-located block in the list 1 reference picture. The list 0 and list 1 motion vectors for direct mode macroblocks are calculated differently depending on whether picture_struct and the reference indicate fields or frames. Also if the list 1 co-located macroblock is an intra-coded block, the motion vectors are set to zero, and the list 0 reference picture for the direct mode is the most recent temporally preceding stored picture.

With possible adaptive switching of frame/field coding at picture level, a B frame or its list 1 reference frame can be coded in either frame structure or field structure. Hence, there are four possible combinations of frame or field coding for a pair of a macroblock in the current B picture and its co-located macroblock in the list 1 reference picture. Calculations of the two motion vectors in direct mode are slightly different for the four cases.

Case 1: Both the current macroblock and its co-located in the list 1 reference picture are in frame mode, as shown in Figure 10-3. The list 0 reference for each block within the current macroblock is the same as the list 0 reference of the co-located block in the list 1 reference picture. Two motion vectors (MV_F, MV_B) are calculated by

$$Z = (TD_B \times 256) / TD_D \qquad MV_F = (Z \times MV + 128) >> 8$$
$$W = Z - 256 \qquad MV_B = (W \times MV + 128) >> 8$$

where TD_B is the temporal distance between the current B frame and the list 0 reference frame, and TD_D is the temporal distance between the list 1 reference frame and the list 0 reference frame (see Figure 10-3).

In the case that the co-located block in the list 1 reference frame has a list 0 motion vector pointing to a long-term frame, the list 0 and list 1 motion vectors for the direct mode current block are calculated by

$$MV_F = MV$$
$$MV_B = 0$$

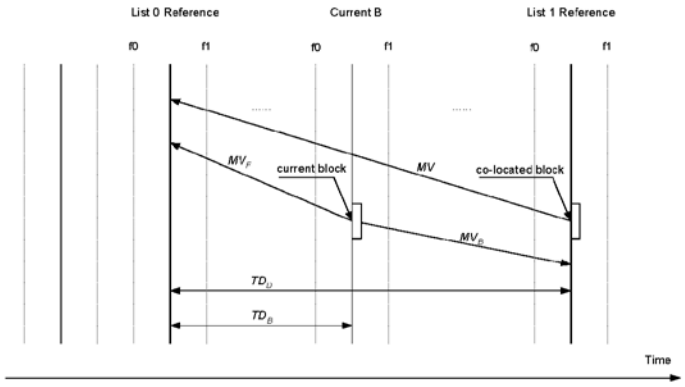

Figure 10-3 – Both the current block and its co-located block in the list 1 reference picture are in frame mode (f0 and f1 indicate the corresponding fields)

Case 2: Both the current macroblock and its co-located macroblock in the list 1 reference picture are in field mode. Two motion vectors for each block within the current macroblock are calculated from the list 0 motion vector of the co-located block in the list 1 reference field of the same parity.

For field 0, the list 0 motion vector of the co-located block will always point to one of the previously coded list 0 fields, as shown in Figure 10-4. The list 0 reference field for the direct mode block will be the same as the list 0 field of the co-located list 1 block, and the list 1 reference field will be field 0 of the list 1 reference frame. The list 0 and list 1 motion vectors ($MV_{F,i}$, $MV_{B,i}$) for the direct mode block are calculated as follows:

$$Z = TD_{B,i} \times 256 / TD_{D,i} \qquad MV_{F,i} = (Z \times MV_i + 128) >> 8$$

$$W = Z - 256 \qquad MV_{B,i} = (W \times MV_i - 128) >> 8$$

where the subscript i is the field index (0 for the 1st field and 1 for the 2nd field), and MV_i is the list 0 motion vector of the co-located block in field i of the list 1 reference frame. $TD_{B,i}$ is the temporal distance between the current B field and the list 0 reference field. $TD_{D,i}$ is the temporal distance between the list 1 reference field and the list 0 reference field.

For field 1, the list 0 motion vector of the co-located list 1 block may point to one of the temporally previous coded fields, in which case calculation of the list 0 and list 1 motion vectors follows the same equations as above.

However, it is also possible that the list 0 motion vector of the co-located block in field 1 of the list 1 reference frame points to field 0 of the same frame, as shown in Figure 10-5. In this case, the two motion vectors for field 1 of the direct mode current block are calculated as follows:

$$Z = -TD_{B,1} \times 256 / TD_{D,1} \qquad MV_{F,1} = (Z \times MV_1 + 128) >> 8$$

$$W = Z - 256 \qquad MV_{B,1} = (W \times MV_1 + 128) >> 8$$

Note that both motion vectors are now pointing to field 0 and field 1 of the list 1 reference frame respectively.

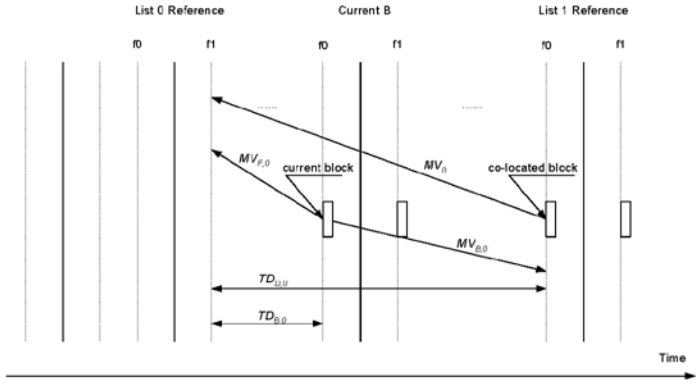

Figure 10-4 – Both the current macroblock and its co-located macroblock in the temporally subsequent picture are in field mode.

In the case that the co-located block in the list 1 reference field has a list 0 motion vector pointing to a long-term field, the list 0 and list 1 motion vectors for the direct mode are calculated by

$MV_{F,i} = MV_i$

$MV_{B,i} = 0$

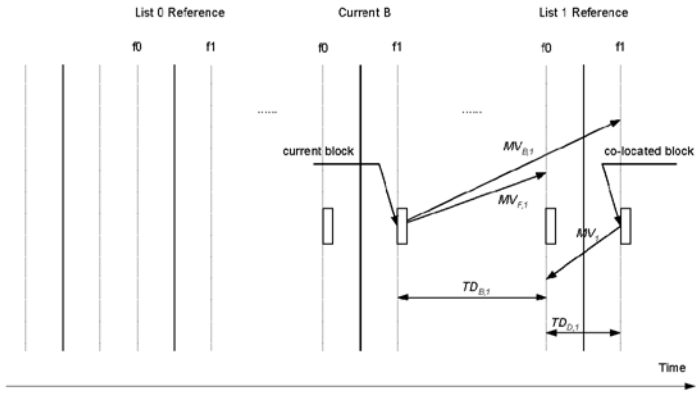

Figure 10-5 – The list 0 motion vector of the co-located block in field 1 of the list 1 reference frame may point to field 0 of the same frame.

Case 3: The current macroblock is in field mode and its co-located macroblock in the list 1 reference picture is in frame mode, as shown in Figure 10-6. Let $y_{current}$ and $y_{co\text{-}located}$ be the vertical indices of the blocks in the current macroblock and in its co-located macroblock respectively. Then, $y_{co\text{-}located} = 2 \times y_{current}$. The blocks in the current macroblock and its co-located macroblock have the same horizontal indices. The list 0 reference field is the same-parity field of the list 0 frame, and the list 1 reference field will be the same-parity field of the list 1 reference frame as shown in Figure 10-6.

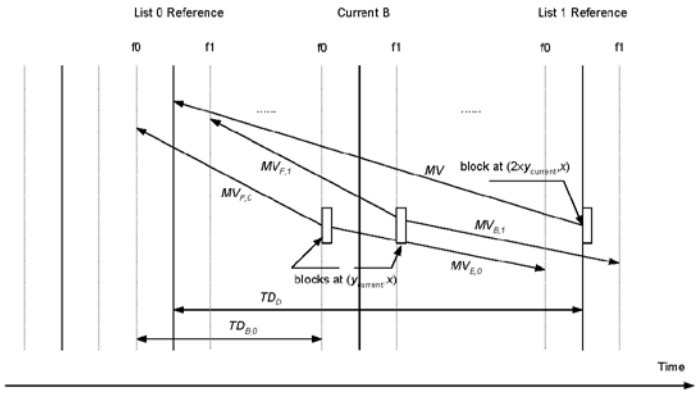

Figure 10-6 – The current macroblock is in field mode and its co-located macroblock in the list 1 reference picture is in frame mode

The motion vectors of a direct mode block are calculated from the list 0 motion vector of the co-located block in the list 1 reference frame as follows:

$$Z = TD_{B,i} \times 256 / TD_D \qquad MV_{F,i} = (Z \times MV + 128) >> 8$$

$$W = Z - 256 \qquad MV_{B,i} = (W \times MV + 128) >> 8$$

In the case that the block in the list 1 reference frame used for the direct mode motion vector calculation has a list 0 motion vector pointing to a long-term picture, the list 0 and list 1 motion vectors for the direct mode are calculated by

$$MV_{F,i} = MV$$
$$MV_{B,i} = 0$$

Case 4: The current macroblock is in frame mode while its co-located macroblock in the list 1 reference picture is in field mode, as shown in Figure 10-7. Let $y_{current}$ and $y_{co\text{-}located}$ be the vertical indices of the blocks in the current macroblock and in its co-located macroblock respectively. Then, $y_{co\text{-}located} = y_{current} / 2$. The blocks in the current macroblock and in its co-located macroblock have the same horizontal indices. The two fields of the co-located block in the list 1 reference may be coded in different modes. Since field 0 of the list 1 reference is temporally close to the current B picture, it is used in calculating the motion vectors and determining the references for direct mode current blocks as shown in Figure 10-7.

Two frame-based motion vectors of direct mode block are calculated as follows:

$$Z = TD_B \times 256 / TD_{D,0} \qquad MV_F = (Z \times MV_0 + 128) >> 8$$

$$W = Z - 256 \qquad MV_B = (W \times MV_0 + 128) >> 8$$

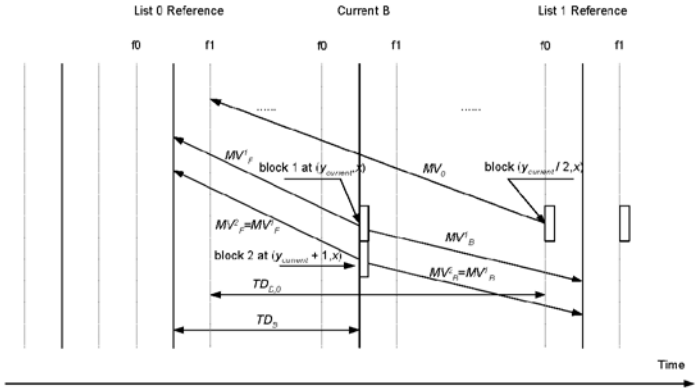

Figure 10-7 – The current macroblock is in frame mode while its co-located macroblock in the list 1 reference picture is in field mode.

In the case that the block in the list 1 reference field used for the direct mode motion vector calculation has a list 0 motion vector pointing to a long-term picture, the list 0 and list 1 motion vectors for the direct mode are calculated by

$$MV_F = MV_0$$
$$MV_B = 0$$

Ed.Note: I don't understand the following sentence. More explanation maybe needed here. This appears to indicate that if the co-located macroblock is in field, then also field mode should be used for the current macroblock/block. Considering though that AFF is done in pairs of macroblocks, is somehow the whole group forced to be in the same mode as its co-located? This is what this seems to indicate.] When mb_frame_field_adaptive_flag == 1, the current direct-mode macroblock is in the same frame/field coding mode as its co-located macroblock.

10.4 Weighted prediction signal generation procedure

If weighted_pred_flag is equal to one, explicit weighted prediction is applied to P and SP slices. If weighted_pred_explicit_flag is equal to one, explicit weighted bi-prediction is applied to B slices. If weighted_bipred_implicit_flag is equal to one, implicit weighted bi-prediction is applied to B slices.

~~If weighted_pred_explicit_flag is equal to one, explicit weighted prediction is applied to P, SP, and B slices. If weighted_bipred_implicit_flag is equal to one, implicit weighted prediction is applied to B slices.~~

10.4.1 Weighted prediction in P and SP slices

In P and SP slices, when weighted_pred_flag is equal to one, weighted prediction is applied for predicted macroblocks. When $0 \le$ mb_type ≤ 4, the luma prediction is generated as

$$P = \text{clip1}((P_0 \times W_0 + 2^{LWD-1}) >> LWD + O_0)$$

and the chroma prediction block is generated as

$$P = \text{clip1}(128 + ((CP_0 - 128) \times CW_0 + 2^{CWD-1}) >> CWD + CO_0)$$

where

P_0 = reference prediction block
LWD = luma_log_weight_denom
W_0 = luma_weight_l0[ref_idx_l0]
O_0 = luma_offset_l0[ref_idx_l0]

Formatted: Lowered by 6 pt
Formatted: Equation, Indent: Left: 1"
Formatted: Lowered by 6 pt
Formatted: Equation, Indent: Left: 1"
Formatted: enumlev1

i = 0 for Cb and 1 for Cr

CP_0 = chroma reference prediction block

CWD = chroma_log_weight_denom

CW_0 = chroma_weight_l0[ref_idx_l0][i]

CO_0 = chroma_offset_l0[ref_idx_l0][i]

To limit the calculation to 16-bit precision, the following conditions shall be met:

$$-128 \le W_0 \le 127$$

$$-128 \le CW_0 \le 127$$

10.4.1 2 Explicit weighted bi-prediction in B slices~~Explicit prediction~~

In B slices, when weighted_bipred_explicit_flag is equal to one, weighted prediction is applied for predicted macroblocks. For reference list 0 prediction, the luma prediction is generated as

$$P = \text{clip1}((P_0 \times W_0 + 2^{LWD-1}) >> LWD + O_0)$$

For reference list 1 prediction, the luma prediction block is generated as:

$$P = \text{clip1}((P_1 \times W_1 + 2^{LWD-1}) >> LWD + O_1)$$

where

P_0 = reference prediction block from list 0

P_1 = reference prediction block from list 1

LWD = luma_log_weight_denom

W_0 = luma_weight_l0[ref_idx_l0]

W_1 = luma_weight_l1[ref_idx_l1]

O_0 = luma_offset_l0[ref_idx_l0]

O_1 = luma_offset_l1[ref_idx_l1]

When bi-prediction is used, the luma prediction block is generated as:

$$P = \text{clip1}((P_0 \times BDW_0 + P_1 \times BDW_1 + 2^{LWD}) >> (LWD + 1) + BDO)$$

where

BDW_0 = luma_weight_bipred_l0[ref_idx_l0][ref_idx_l1]

BDW_1 = luma_weight_bipred_l1[ref_idx_l0][ref_idx_l1]

BDO = luma_offset_bipred[ref_idx_l0][ref_idx_l1]

For reference list 0 prediction, the chroma prediction block is generated as

$$P = \text{clip1}(128 + ((CP_0 - 128) \times CW_0 + 2^{CWD-1}) >> CWD + CO_0)$$

For reference list 1 prediction, the chroma prediction block is generated as:

$$P = \text{clip1}(128 + ((CP_1 - 128) \times CW_1 + 2^{CWD-1}) >> CWD + CO_1)$$

When bi-prediction is used, the chroma prediction block is generated as

$$P = \text{clip1}(128 + ((CP_0 - 128) \times CBDW_0 + (CP_1 - 128) \times CBDW_1 + 2^{CWD}) >> (CWD + 1) + CBDO)$$

Formatted: enumlev1, Space Before: 0 pt

Formatted: enumlev1

Formatted: Indent: Left: 1", Space Before: 9.65 pt, After: 12 pt

Formatted: Equation, Indent: Left: 1"

Formatted: Equation, Indent: Left: 1", Space Before: 0 pt

Formatted: enumlev1

Formatted: Indent: Left: 1"

Formatted: Equation, Indent: Left: 1"

Formatted: Equation, Indent: Left: 1"

Formatted: Indent: Left: 0.5"

where

i = 0 for Cb and 1 for Cr

CP_0 = chroma reference prediction block from list 0

CP_1 = chroma reference prediction block from list 1

CWD = chroma_log_weight_denom

CW_0 = chroma_weight_l0[ref_idx_l0][i]

CW_1 = chroma_weight_l1[ref_idx_l1][i]

CO_0 = chroma_offset_l0[ref_idx_l0][i]

CO_1 = chroma_offset_l1[ref_idx_l1][i]

$CBDW_0$ = chroma_weight_bipred_l0[ref_idx_l0][ref_idx_l1][i]

$CBDW_1$ = chroma_weight_bipred_l1[ref_idx_l0][ref_idx_l1][i]

$CBDO$ = chroma_offset_bipred[ref_idx_l0][ref_idx_l1][i]

To limit the calculation to 16-bit precision, the following conditions shall be met:

$$-128 \le W_0 \le 127$$

$$-128 \le W_1 \le 127$$

$$-128 \le W_0 + W_1 \le 127$$

$$-128 \le CW_0 \le 127$$

$$-128 \le CW_1 \le 127$$

$$-128 \le CW_0 + CW_1 \le 127$$

$$-128 \le BDW_0 + BDW_1 \le 127$$

$$-128 \le CBDW_0 + CBDW_1 \le 127$$

~~In P, SP and B slices, when weighted_pred_explicit_flag is equal to one, weighted prediction is applied for predicted macroblocks.~~

~~For P and SP slices when 0 ≤ mb_type ≤ 4 and for B slices when Pred_L0 is used for prediction, the luma prediction is generated as~~

~~$P = \text{clip1}((P_1 \times W1 + 2^{LWD-1}) >> LWD + O1)$~~

~~For B slices when Pred_L1 is used for prediction, the luma prediction block is generated as:~~

~~$P = \text{clip1}((P_2 \times W2 + 2^{LWD-1}) >> LWD + O2)$~~

~~where~~

~~P_1 = reference prediction block~~

~~P_2 = second reference prediction block~~

~~LWD = luma_log_weight_denom~~

~~$W1$ = luma_weight_l0[ref_idx_l0]~~

~~$W2$ = luma_weight_l1[ref_idx_l1]~~

~~$O1$ = luma_offset_l0[ref_idx_l0]~~

Formatted: enumlev1, Space Before: 0 pt

Formatted: enumlev1

Formatted: Indent: Left: 1", Space Before: 9.65 pt, After: 12 pt

$O2$ = luma_offset_l1[ref_idx_l1]

For B slices when bi-prediction is used, the luma prediction block is generated as:

$P = \text{clip1}((P_1 \times BDW1 + P_2 \times BDW2 + 2^{LWD}) >> (LWD + 1) + BDO)$

where

$BDW1$ = luma_weight_bipred_l0[ref_idx_l0][ref_idx_l1]

$BDW2$ = luma_weight_bipred_l1[ref_idx_l0][ref_idx_l1]

BDO = luma_offset_bipred[ref_idx_l0][ref_idx_l1]

For P and SP slices when 0 ≤ mb_type ≤ 4 and for B slices when Pred_L0 is used for prediction, the chroma prediction block is generated as

$P = \text{clip1}(128 + ((CP_1 - 128) \times CW1 + 2^{CWD-1}) >> CWD + CO1)$.

For B slices when Pred_L1 is used for prediction, the chroma prediction block is generated as:

$P = \text{clip1}(128 + ((CP_2 - 128) \times CW2 + 2^{CWD-1}) >> CWD + CO2)$.

For B slices when bi-prediction is used, the chroma prediction block is generated as

$P = \text{clip1}(128 + ((CP_1 - 128) \times CBDW1 + (CP_2 - 128) \times CBDW2 + 2^{CWD}) >> (CWD + 1) + CBDO)$.

where

j = 0 for Cb and 1 for Cr

CP_1 = first chroma reference prediction block

CP_2 = second chroma reference prediction block

CWD = chroma_log_weight_denom

$CW1$ = chroma_weight_l0[ref_idx_l0][j]

$CW2$ = chroma_weight_l1[ref_idx_l1][j]

$CO1$ = chroma_offset_l0[ref_idx_l0][j]

$CO2$ = chroma_offset_l1[ref_idx_l1][j]

$CBDW1$ = chroma_weight_bipred_l0[ref_idx_l0][ref_idx_l1][j]

$CBDW2$ = chroma_weight_bipred_l1[ref_idx_l0][ref_idx_l1][j]

$CBDO$ = chroma_offset_bipred[ref_idx_l0][ref_idx_l1][j]

To limit the calculation to 16-bit precision, the following conditions shall be met:

$-128 \le W1 \le 127$

$-128 \le W2 \le 127$

$-128 \le W1 + W2 \le 127$

$-128 \le BD1 + BD2 \le 127$

$-128 \le CW1 \le 127$

$-128 \le CW2 \le 127$

~~$-128 \le CW1 + CW2 \le 127$~~

~~$-128 \le CBD1 + CBD2 \le 127$~~

10.4.~~2~~3 Implicit ~~Bipredictive~~ bi-predictive ~~Weighting~~weighting

When weighted_bipred_implicit_flag is equal to 1, the prediction weighting factors are not sent explicitly and the luma and chroma predictions are generated as follows.

If the decoding order of the reference picture indicated by ref_idx_l0 is previous to or the same as that indicated by ref_idx_l1, or for skipped macroblocks or direct mode, the prediction signals are generated as follows:

$$P = \text{clip1}(P_0 \times 2 - P_1)$$

where

P_0 = reference prediction block from list 0

P_1 = reference prediction block from list 1;

otherwise, the prediction signals are generated as follows

$$P = (P_0 + P_1 + 1) >> 1$$

where

P_0 = reference prediction block from list 0

P_1 = reference prediction block from list 1.

~~In the implicit bipredictive weighting mode, when weighted_bipred_implicit_flag is equal to 1, the prediction weighting factors are not sent explicitly and are instead inferred as follows:~~

~~**luma_constant_bipred_factor**[ref_idx_l0][ref_idx_l1] = 0~~

~~**chroma_constant_bipred_factor**[ref_idx_l0][ref_idx_l1] = 0~~

~~If the decoding order of the reference picture indicated by ref_idx_l0 is less than or equal to that indicated by ref_idx_l1, or for skipped macroblocks or direct mode, the following parameters are used:~~

~~**luma_weight_bipred_factor1**[ref_idx_l0][ref_idx_l1] = 2~~

~~**luma_weight_bipred_factor2**[ref_idx_l0][ref_idx_l1] = 1~~

~~**luma_logarithmic_weight_denominator** = 1~~

~~**chroma_weight_bipred_factor1**[ref_idx_l0][ref_idx_l1] = 2~~

~~**chroma_weight_bipred_factor2**[ref_idx_l0][ref_idx_l1] = 1~~

~~**chroma_logarithmic_weight_denominator** = 1~~

~~Otherwise, the following parameters are used:~~

~~**luma_weight_bipred_factor1**[ref_idx_l0][ref_idx_l1] = 1~~

~~**luma_weight_bipred_factor2**[ref_idx_l0][ref_idx_l1] = 1~~

~~**luma_logarithmic_weight_denominator** = 0~~

~~**chroma_weight_bipred_factor1**[ref_idx_l0][ref_idx_l1] = 1~~

~~**chroma_weight_bipred_factor2**[ref_idx_l0][ref_idx_l1] = 1~~

~~**chroma_logarithmic_weight_denominator** = 0~~

~~Implicit bi-predictive weighting mode applies only to bi-predictive blocks.~~

~~For B slices when bi-prediction is used, the luma prediction block is generated as:~~

Formatted: Lowered by 6 pt

Formatted: Equation, Indent: Left: 1", Space After: 0 pt

Formatted: Lowered by 6 pt

Formatted: Equation, Indent: Left: 1", Space After: 0 pt

Formatted: enumlev1

~~$P = \text{clip1}((P_1 \times BDW1 + P_2 \times BDW2 + 2^{LWD}) >> (LWD + 1) + BDO)$~~

~~Where~~

~~P_1 = reference prediction block~~

~~P_2 = second reference prediction block~~

~~LWD = luma_log_weight_denom~~

~~$BDW1$ = luma_weight_bipred_l0[ref_idx_l0][ref_idx_l1]~~

~~$BDW2$ = luma_weight_bipred_l1[ref_idx_l0][ref_idx_l1]~~

~~BDO = luma_offset_bipred[ref_idx_l0][ref_idx_l1]~~

~~For B slices when bi-prediction is used, the chroma prediction block is generated as~~

~~$P = \text{clip1}(128 + ((CP_1 - 128) \times CBDW1 + (CP_2 - 128) \times CBDW2 + 2^{CWD}) >> (CWD + 1) + CBDO)$,~~

~~where~~

~~j = 0 for Cb and 1 for Cr~~

~~CP_1 = first chroma reference prediction block~~

~~CP_2 = second chroma reference prediction block~~

~~CWD = chroma_log_weight_denom~~

~~$CBDW1$ = chroma_weight_bipred_l0[ref_idx_l0][ref_idx_l1]~~

~~$CBDW2$ = chroma_weight_bipred_l1[ref_idx_l0][ref_idx_l1]~~

~~$CBDO$ = chroma_offset_bipred[ref_idx_l0][ref_idx_l1]~~

11 Decoding process for SP and SI slices

~~[Ed.Note: Reference to tables and equations etc need to be fixed.]~~

11.1 ~~Introduction~~General

SP slices make use of motion-compensated predictive coding to exploit temporal redundancy in the sequence, in a similar manner to P slices. SI slices make use of spatial prediction, in a similar manner to I slices. Unlike P slices, however, SP slice coding allows identical reconstruction of a slice even when different reference pictures are being used. SI slice coding allows identical reconstruction to a corresponding SP slice.

NOTE - Above mentioned properties of SP and SI slices provide functionalities for bitstream switching, splicing, random access, VCR functionalities such as fast-forward, and error resilience/recovery.

An SP slice consists of macroblocks coded either as Intra type (Intra_4x4 or Intra_16x16) or Pred type (MbSkip, Pred_L0_16x16, Pred_L0_L0_16x8, Pred_L0_L0_8x16, Pred_8x8 or Pred_8x8ref0). An SI slice consists of macroblocks coded either as Intra type or SIntra4x4 type.

Intra macroblocks in SP slices shall be decoded as described in subclause 8.3. All other macroblocks, including MbSkip, are decoded as described below.

The Intra_8x8 sub-partition mode shall not be present in SP slices.

Intra macroblocks in SI slices are decoded as described in subclause 8.3, with the addition that the prediction mode of a neighbouring SIntra_4x4 block is considered to be "mode 2: DC prediction". SIntra_4x4 macroblocks are decoded as described below.

11.2 SP decoding process for non-switching pictures

This subclause applies to all macroblocks in SP slices in which sp_for_switch_flag == 0, except those with slice_type() equal to Intra_4x4 or Intra_16x16. It does not apply to SI slices.

Figure ~~12~~11-1 depicts generic decoding for non-intra coded macroblocks in SP slices. The prediction P(x,y) for the current macroblock of the slice being decoded is formed by the motion compensated prediction block using the same process as is used in P slice decoding. After forming the predicted block P(x,y), decoding is performed as follows.

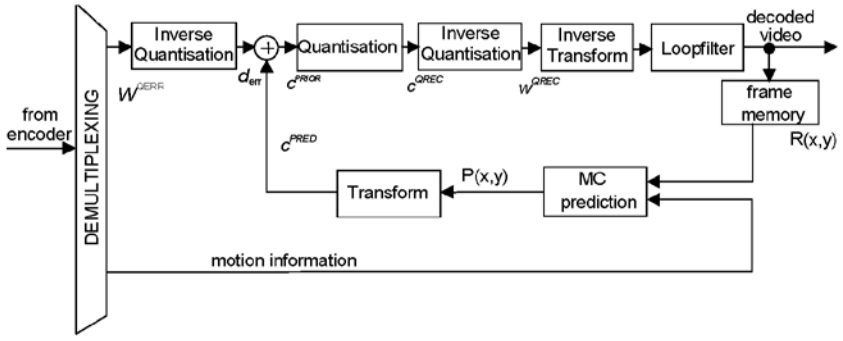

Figure 11-1 – A block diagram of a conceptual decoder for non-intra coded macroblocks in SP slices in which sp_for_switch_flag == 0.

11.2.1 Luma transform coefficient decoding

The predicted block, P(x,y), where P(x,y) ={p_{00} … p_{33}}, is transformed according to Equation 11-1 to produce transform coefficients c^{PRED}.

$$c^{PRED} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \quad (11\text{-}1)$$

The received prediction residual coefficients, w^{QERR}, are scaled using quantisation parameter QP, and added to the transform coefficients of the prediction block, as in Equation 11-2.

$$c_{ij}^{PRIOR} = c_{ij}^{PRED} + (((w_{ij}^{QERR} * R_{ij}^{(QP\%6)} * A_{ij}) << (QP/6)) >> 6) \quad i, j = 0,\ldots,3 \quad (11\text{-}2)$$

where R is defined in Equation 8-~~13~~40, and where A is defined as:

$A_{ij}^{(m)} = 16$ for (i,j) = {(0,0),(0,2),(2,0),(2,2)},

$A_{ij}^{(m)} = 25$ for (i,j) = {(1,1),(1,3),(3,1),(3,3)},

$A_{ij}^{(m)} = 20$ otherwise;.

For luma, QP = QP_Y, as defined in Equation 7-7 and 7-9.

The coefficients $Q_{ij}^{(m)}$, used in the formulas below, are defined as:

$Q_{ij}^{(m)} = M_{m,0}$ for (i,j) = {(0,0),(0,2),(2,0),(2,2)},

$Q_{ij}^{(m)} = M_{m,1}$ for (i,j) = {(1,1),(1,3),(3,1),(3,3)},

$Q_{ij}^{(m)} = M_{m,2}$ otherwise;

where the first and second subscripts of M are row and column indices, respectively, of the matrix defined as:

$$M = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix}$$

The resulting sum, c^{PRICR}, is quantised with a quantisation parameter QS, as in Equation 11-3.

For luma, QS = QS_Y, which is defined in Equation 7-8.

$$c_{ij}^{QREC} = \{Sign(c_{ij}^{PRIOR})*[Abs(c_{ij}^{PRIOR})*Q_{ij}^{(QS\%6)} + (1<<(15+QS/6))]\}>>(16+QS/6) \quad i,j = 0,\ldots,3 \qquad (11\text{-}3)$$

These quantised levels, c^{QREC}, are scaled as in Equation 11-4.

$$w_{ij}^{QREC} = (c_{ij}^{QREC} * R_{ij}^{(QS\%6)}) << (QS/6) \quad i,j = 0,\ldots,3 \qquad (11\text{-}4)$$

where R is defined in Equation 8-~~13~~40.

The transform and reconstruction processes are performed for these scaled levels, as defined in Equations 8-~~22~~48 through 8-~~33~~59. Finally, after the reconstruction of a macroblock, filtering takes place as described in subclause 8.~~6~~7.

11.2.2 Chroma transform coefficient decoding

The decoding of chroma components for non-intra coded macroblocks in SP slices is similar to the decoding of luma components.

The predicted block, P(x,y), where P(x,y) ={p_{00} … p_{33}}, is transformed according to Equation 11-1 to produce transform coefficients c^{PRED}. An additional 2x2 transform is applied to the DC coefficients of these blocks. The 2 dimensional 2x2 transform procedure is defined in Equation 11-5:

$$c^{PRED} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} DC_{00} & DC_{01} \\ DC_{10} & DC_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (11\text{-}5)$$

The received DC prediction residual coefficients, w^{QERR}, are scaled using quantisation parameter QP, and added to the DC transform coefficients of the prediction block, as in Equation 11-6.

$$c_{ij}^{PRICR} = c_{ij}^{PRED} + (((w_{ij}^{QERR} * R_{ij}^{(QP\%6)} * A_{ij}) << (QP/6)) >>5) \quad i,j = 0,\ldots,3 \qquad (11\text{-}6)$$

The resulting sum, c^{PRICR}, is quantised with a quantisation parameter QS, as in Equation 11-7.

$$c_{ij}^{QREC} = \{Sign(c_{ij}^{PRIOR})*[Abs(c_{ij}^{PRIOR})*Q_{ij}^{(QS\%6)} + (1<<(16+QS/6))]\}>>(17+QS/6) \quad i,j = 0,\ldots,3 \qquad (11\text{-}7)$$

These quantised levels, c_{ij}^{QREC}, are scaled as in Equation 11-4.

AC coefficients are decoded in an identical process to that used for luma.

The value of QP to be used for chroma data, denoted QP_C, is obtained from QP_Y using the relationship specified in Table 9-1. Similarly, the value of QS to be used for chroma data, denoted QS_C, is obtained from QS_Y using the relationship specified in Table 9-1.

11.3 SP and SI slice decoding process for switching pictures

This subclause applies to all macroblocks in SP slices in which sp_for_switch_flag == 1, except those with slice_type() equal to Intra_4x4 or Intra_16x16; and to all macroblocks in SI slices, except those with slice_type() equal to Intra_4x4 or Intra_16x16.

Figure ~~12~~11-2 depicts generic decoding for macroblocks in SI and SP slices that are not Intra coded. In SP slices, the prediction P(x,y) for the current coded macroblock of the slice being decoded is formed by the inter prediction block

using the same process as is used in P slice decoding. In SI slices, the prediction P(x,y) is formed by intra prediction using the same process as is used in I slice decoding. After forming the predicted block P(x,y), the decoding of SI and non-intra coded macroblocks in SP slices follows the same steps.

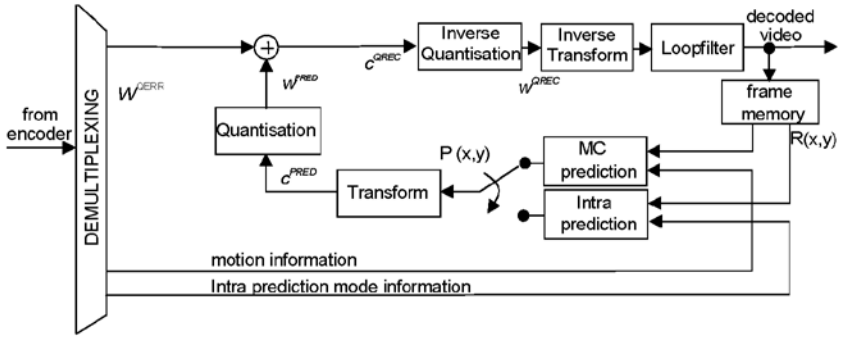

Figure 11-2 – A block diagram of a conceptual decoder for non-intra macroblocks in SI slices; and for non-intra coded macroblocks in SP slices in which sp_for_switch_flag == 1.

When decoding SIntra_4x4 macroblocks, the intra prediction modes of the neighbouring SIntra_4x4 and Intra_4x4 blocks are taken into account as described in subclause ~~9.48.5~~.

When decoding Intra_4x4 macroblocks, the intra prediction modes of the neighbouring Intra_4x4 blocks are taken into account as described in subclause ~~9.48.5~~, but the prediction mode of the neighbouring SIntra_4x4 blocks are considered to be "mode 2: DC prediction".

11.3.1 Luma transform coefficient decoding

The predicted block, P(x,y), where P(x,y) ={p_{00} … p_{33}}, is transformed according to Equation 11-1 to produce transform coefficients c^{PRED}. These transform coefficients are then quantised with a quantisation parameter QS, as in Equation 11-8.

$$w_{ij}^{PRED} = \{Sign(c_{ij}^{PRED})*[Abs(c_{ij}^{PRED})*Q_{ij}^{(QS\%6)} + (1<<(15+QS/6))]\}>>(16+QS/6) \quad i, j = 0,\ldots, 3 \qquad (11\text{-}8)$$

Note: Equation 11-8 is the same as Equation 11-3 except for the change of name of input and output variables.

For luma, QS = QS_Y, which is defined in Equation 7-8.

The received prediction residual coefficients, w^{QERR}, are added to these quantised transform coefficients of the prediction block, as in Equation 11-9.

$$c_{ij}^{QREC} = w_{ij}^{PRED} + w_{ij}^{QERR} \quad i, j = 0,\ldots, 3 \qquad (11\text{-}9)$$

These quantised levels, c^{QREC}, are decoded as in subclause 11.1.1

11.3.1.2 Chroma transform coefficient decoding

The decoding of chroma components for SP and SI non-intra macroblocks is similar to the decoding of luma components.

The predicted block, P(x,y), where P(x,y) ={p_{00} … p_{33}}, is transformed according to Equation 11-1 to produce transform coefficients c^{PRED}. An additional 2x2 transform is applied to the DC coefficients of these blocks as in Equation 11-5:

The DC transform coefficients are then quantised with a quantisation parameter QS, as in Equation 11-10.

$$w_{ij}^{PRED} = \{Sign(c_{ij}^{PRED})*[Abs(c_{ij}^{PRED})*Q_{ij}^{(QS\%6)} + (1<<(16+QS/6))]\}>>(17-QS/6) \quad i, j = 0,\ldots, 3 \qquad (11\text{-}10)$$

Formatted: Not Highlight

Formatted: Not Highlight

Formatted: Not Superscript/ Subscript

Formatted: Not Superscript/ Subscript

Formatted: Indent: Left: 1", Hanging: 0.5"

NOTE - Equation 11-10 is the same as Equation 11-7 except for the change of name of input and output variables.

The received prediction residual DC coefficients, w^{QERR}, are added to these quantised DC transform coefficients of the prediction block, as in Equation 11-9.

AC coefficients are decoded in an identical process to that used for luma.

The value of QP to be used for chroma data, denoted QP_C, is obtained from QP_Y using the relationship specified in Table 9-1. Similarly, the value of QS to be used for chroma data, denoted QS_C, is obtained from QS_Y using the relationship specified in Table 9-1.

12 Adaptive block size transforms

12.1 Introduction

In this clause, the modifications to the syntax and semantics in clause 7 and the changes to the decoding process in clause 8 and to entropy coding in clause 9 for adaptive block size transforms are described.

If adaptive_block_size_transform_flag − − 1, additional transforms of size 4x8, 8x4, and 8x8 are specified for the luma residual. The chroma residual decoding process remains unchanged. Adaptive block size transforms are used for all macroblocks with $QP_Y >= 12$. In inter predicted macroblocks, the transform block size is indicated by the block size used for inter prediction. For intra macroblocks, the block size used for intra prediction is connected to the block size of the transformation. For intra macroblocks in inter slices, the block size is indicated by the syntax element intra_block_typeABT. For intra slices, the intra block size is indicated by the macroblock block mode.

Figure 12-1 shows the order of the assignments of syntax elements for luma resulting from coding a macroblock to sub-blocks of the macroblock if the ABT features is used. The assignment of blocks and coded_block_patternY is specified in Figure 12-1. An 8x8 block may contain 1, 2, or 4 transform blocks. An indication that an 8x8 block contains coefficients means that the 8x8 transform blocks or one or more of the 2, or 4 transform blocks within the 8x8 block contains coefficients. The chroma 4x4 residual blocks are ordered after the luma blocks as indicated in Figure 6-6.

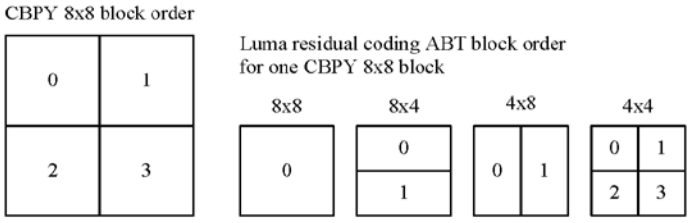

Figure 12-1 – Ordering of blocks for CBPY and luma residual coding of ABT blocks

12.2 ABT Syntax

12.2.1 Macroblock layer syntax

macroblock_layer_abt( ) {	Category	Descriptor
mb_type	4	ue(v) | ae(v)
if(num_mb_partition[mb_type] == 4)		
sub_mb_pred_abt(mb_type)	4	
else		
mb_pred_abt(mb_type)	4	
SendResidual = 0		
if(mb_partition_pred_mode(, 1) == Intra && mb_type != Intra_4x4) { /* Intra_16x16_X_Y_Z mb_type */		
coded_block_pattern	4	me(v) | ae(v)
if(coded_block_pattern > 0) ~~{~~		
SendResidual = 1		
~~**delta_qp**~~	~~4~~	~~se(v) | ae(v)~~
~~residual_abt( )~~	~~5 | 6~~	
~~}~~		
} else {		
coded_block_pattern	4	me(v) | ae(v)
if(coded_block_pattern > 0)		
SendResidual = 1		
}		
if(SendResidual) {		
if(!mb_frame_field_adaptive_flag || (mb_frame_field_adaptive_flag && (pic_structure == 0 || pic_structure == 3 || pic_structure == 4) && first_non_skip_mb_in_pair( ))		
delta_qp	4	se(v) | ae(v)
residual_abt( )	5 | 6	
}		
}		

~~[Ed. Note: Do not send delta_qp twice in the case of MB level AFF. AFF open issue.]~~

12.2.1.1 Macroblock prediction syntax

mb_pred_abt(mb_type) {	Category	Descriptor
if(mb_partition_pred_mode(, 1) == Intra) {		

if(mb_type == Intra_4x4
mb_type == ABTIntra_4x4 ||
mb_type == ABTIntra_4x8 ||
mb_type == ABTIntra_8x4 ||
mb_type == ABTIntra_8x8) {		

if(MbABTFlag && (slice_type( ) == Pred || slice_type( ) == BiPred)		
intra_block_typeABT	4	me(v) | ae(v)
for(i = 0; i < num_mb_intra_partition( ); i++) /* for each luma block */		
intra_pred_mode	4	ce(v) | ae(v)
}		
intra_chroma_pred_mode	4	ue(v) | ae(v)
} else if(mb_type != Direct_16x16) {		
for(i = 0; i < num_mb_partition(mb_type, i); i++)		
if(num_ref_idx_l0_active_minus1 > 0 && mb_partition_pred_mode(mb_type, i) != Pred_L1)		
ref_idx_l0	4	ue(v) | ae(v)
for(i = 0; i < num_mb_partition(mb_type, i); i++) {		
if(num_ref_idx_l1_active_minus1 > 0 && mb_partition_pred_mode(mb_type, i) != Pred_L0)		
ref_idx_l1	4	ue(v) | ae(v)
for(i = 0; i < num_mb_partition(mb_type); i++) {		
if(mb_partition_pred_mode (mb_type, i) != Pred_L1)		
for(j = 0; j < 2; j++)		
mvd_l0[i][j]	4	se(v) | ae(v)
for(i = 0; i < num_mb_partition(mb_type); i++) {		
if(mb_partition_pred_mode(mb_type, i) != Pred_L0)		
for(j = 0; j < 2; j++)		
mvd_l1[i][j]	4	se(v) | ae(v)
}		
}		

12.2.1.2 Sub macroblock prediction syntax

sub_mb_pred_abt(mb_type) {	Category	Descriptor
for(i = 0; i < 4; i++) /* for each sub macroblock */		
sub_mb_type[i]	4	ue(v) | ae(v)
IntraChromaPredModeFlag = 0		
for(i = 0; i < 4; i++) /* for each sub macroblock */		
if(sub_mb_type[i] == Intra_8x8) {		
if(MbABTFlag) {		
intra_block_typeABT	4	me(v) | ae(v)
for(j = 0; j < num_sub_mb_intra_partition( ); j++) {		
intra_pred_mode	4	ce(v) | ae(v)
IntraChromaPredModeFlag = 1		
}		
}		
if(IntraChromaPredModeFlag)		
intra_chroma_pred_mode	4	ue(v) | ae(v)
for(i = 0; i < 4; i++) /* for each sub macroblock */		
if(num_ref_idx_l0_active_minus1 > 0 && mb_type != Pred_8x8ref0 && sub_mb_type[i] != Intra_8x8 && sub_mb_type[i] != Direct_8x8 && sub_mb_pred_mode(sub_mb_type[i]) != Pred_L1)		
ref_idx_l0	4	ue(v) | ae(v)
for(i = 0; i < 4; i++) /* for each sub macroblock */		
if(num_ref_idx_l1_active_minus1 > 0 && (sub_mb_type[i] != Intra_8x8 && sub_mb_type[i] != Direct_8x8 && sub_mb_pred_mode(sub_mb_type[i]) != Pred_L0)		
ref_idx_l1	4	ue(v) | ae(v)
for(i = 0; i < 4; i++) /* for each sub macroblock */		
if(sub_mb_type[i] != Intra_8x8 && sub_mb_type[i] != Direct_8x8 && sub_mb_pred_mode(sub_mb_type[i]) != Pred_L1)		
for(j = 0; j < num_sub_mb_partition(sub_mb_type[i]); j++)		
for(k = 0; k < 2; k++)		
mvd_l0[i][j][k]	4	se(v) | ae(v)
for(i = 0; i < 4; i++) /* for each sub macroblock */		
if(sub_mb_type[i] != Intra_8x8 && sub_mb_type[i] != Direct_8x8 && sub_mb_pred_mode(sub_mb_type[i]) != Pred_L0)		
for(j = 0; j < num_sub_mb_partition(sub_mb_type[i]); j++)		
for(k = 0; k < 2; k++)		
mvd_l1[i][j][k]	4	se(v) | ae(v)
}		

12.2.1.3 Residual data syntax

residual_abt(mb_type) {	Category	Descriptor
if(entropy_coding_mode == 1) {		
residual_4x4block = residual_4x4block_cabac( ) /* function pointer */	5 | 6	
residual_subblock = residual_subblock_cabac( ) /* function pointer */	5 | 6	
} else {		
residual_4x4block = residual_4x4block_cavlc( ) /* function pointer */	5 | 6	
residual_subblock = residual_subblock_cavlc( ) /* function pointer */	5 | 6	
}		
if(mb_type == Intra_16x16)		
residual_4x4block(intra16x16DC, 16)	5	
for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
for(i4x4 = 0; i4x4 < num_sub_blocks( ); i4x4++) /* each sub-block of block */		
if(coded_block_pattern & (1 << i8x8))		
if(mb_type = = Intra_16x16)		
residual_4x4block(intra16x16AC, 16)	5	
else		
if(MbABTFlag && (slice_type() == Intra || mb_type == Intra_4x4)		
residual_subblock(IntraABT, sub_block_type)	5 | 6	
else		
residual_subblock(InterABT, sub_block_type)	5 | 6	
if(coded_block_pattern & 0x30) /* chroma DC residual coded */		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
residual_4x4block(chromaDC, 4))	5 | 6	
if(coded_block_pattern & 0x20) /* chroma AC residual coded */		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
For(i4x4 = 0; i4x4 < 4; i4x4++)		
residual_4x4block(chromaAC, 16)	5 | 6	
}		

~~[Ed.Note: mwi to Detlev: might want to rename InterABT and IntraABT?]~~

12.2.1.3.1 Residual sub block CAVLC syntax

residual_subblock_cavlc(block_coding_type, sub_block_type) {		
if(! MbABTFlag)		
residual_4x4block_cavlc(luma, 16)	5 | 6	
else {		
if(block_coding_type == IntraABT) {		
~~num_coeffABT~~num_coeff_abt	5 | 6	ce(v)
for (i=0; i<~~num_coeffABT~~num_coeff_abt; i++) {		
code_number	5 | 6	ce(v)
if(code_number != escape) {		
level[i] = code_number2level_intra(code_number, sub_block_type)		
run[i] = code_number2run_intra(code_number, sub_block_type)		
} else {		
code_number	5 | 6	ce(v)
level[i] = escape_level[code_number]		
code_number	5 | 6	ce(v)
run[i] = escape_run[code_number]		
}		
}		
} else {		
for(i=0; i<max_numcoeffABT(sub_block_type); i++) {		
code_number	5 | 6	ce(v)
if(code_number == 0)		
break;		
if(code_number != escape) {		
run[i] = code_number2run_inter(code_number, sub_block_type)		
level[i] = code_number2level_inter(code_number, sub_block_type)		
} else {		
code_number	5 | 6	ce(v)
level[i] = escape_level[code_number]		
code_number	5 | 6	ce(v)
run[i] = escape_run[code_number]		
}		
}		
}		
}		
}		

12.2.1.3.2 Residual sub block CABAC syntax

residual_subblock_cabac(block_coding_type, sub_block_type) {	Category	Descriptor
if(! MbABTFlag)		
residual_4x4block_cabac(luma, 16)	5 | 6	
else {		
if(sub_block_type != 8x8)		
cbp4	5 | 6	ac(v)
if(cbp4 || sub_block_type == 8x8) {		
max_numcoeff = max_num_coeff_abt(sub_block_type)		
significant_coeff[max_numcoeff - 1] = 1		
for(i = 0; i < max_numcoeff; i++) {		
significant_coeff[i]	5 | 6	ae(v)
if(significant_coeff[i] && i < max_numcoeff-1) {		
last_significant_coeff[i]	5 | 6	ae(v)
if(last_significant_coeff[i])		
max_numcoeff = i + 1		
}		
}		
for(i = max_numcoeff-1; i >= 0; i--)		
if(significant_coeff[i]) {		
coeff_absolute_value_minus1[i **]**	5 | 6	ae(v)
coeff_sign[i **]**	5 | 6	ae(v)
}		
}		
}		
}		

12.3 ABT Semantics

12.3.1 Macroblock layer semantics

MbABTFlag indicates the usage of ABT for the macroblock. If adaptive_block_transform_flag is equal 1 and QP_Y >= '12' MbABTFlag = 1 else MbABTFlag = 0.

The meaning of mb_type 'Intra_4x4' in Inter slices is modified for ABT. If MbABTFlag is equal 1, this macroblock type indicates application of ABT Intra prediction and transformation. In Inter slices, the block size used for prediction and transform is indicated by the syntax element intra_block_typeΛBT. For I slices, the prediction and transform block size is derived from the macroblock mode. The modified macroblock types for I slices are specified in Table 12-1 below.

Table 12-1 – Modified macroblock types for I slices

Value of mb_type	Name of mb_type in I slices if MbABTFlag = = 1	mb_partition_pred_ mode(, 1)
0	ABTIntra_4x4	Intra
1	ABTIntra_4x8	Intra
2	ABTIntra_8x4	Intra
3	ABTIntra_8x8	Intra

12.3.1.1 Macroblock prediction semantics

num_mb_intra_partition( ) depends on mb_type in I slices and on intra_block_typeABT in P slices. The assignment of num_mb_intra_partition( ) is specified in Table 12-2.

intra_block_typeABT indicates the blocksize used for ABT intra decoding in Inter slices. Blocks of size 4x4, 4x8, 8x4 and 8x8 samples may be used for ABT intra prediction. Table 12-2 provides num_mb_intra_partition and num_sub_mb_intra_partition specifying the number of intra_pred_mode syntax elements to be decoded.

Table 12-2 – ABT intra partitions

mb_type	intra_block_typeABT	num_mb_intra_partition( )	num_sub_mb_intra_partition( )
ABTIntra_4x4	4x4	16	4
ABTIntra_4x8	4x8	8	2
ABTIntra_8x4	8x4	8	2
ABTIntra_8x8	8x8	4	1

12.3.1.2 Sub macroblock prediction semantics

If MbABTFlag is equal 1, the number of decoded intra prediction modes is indicated by num_sub_mb_intra_partition( ) as specified in Table 12-2.

12.3.1.3 Residual data semantics

num_sub_blocks( ) indicates the number of partitions in a luma 8x8 block. If MbABTFlag is equal 0 num_sub_blocks is 4. If MbABTFlag equals 1, num_sub_blocks may be 1, 2, or 4 depending on sub_block_type as specified in Table 12-3.

Table 12-3 – ABT Intra Block Types

sub_block_type	num_sub_blocks( )	max_num_coeff_abt( )
4x4	4	16
4x8	2	32
8x4	2	32
8x8	1	64

sub_block_type indicates the block type used for decoding of an 8x8 block. The decoded blocks may be of size 4x4, 4x8, 8x4, or 8x8 samples. For intra slices with MbABTFlag == 1, sub_block_type for macroblock type ABTIntra_NxM is NxM. For intra macroblocks and intra sub macroblocks in inter slices sub_block_type is equal to intra_block_typeABT. For inter macroblocks with macroblock type Pred_x_NxM or Pred_x_y_NxM or BiPred_x_NxM or BiPred_x_y_NxM, x, y = L0, L1, Bi, sub_block_type is N'xM'. N' and M' are derived from N and M by clipping:

$$N' = \text{Clip3}(4, 8, N) \quad (12\text{-}1)$$

$$M' = \text{Clip3}(4, 8, M) \quad (12\text{-}2)$$

For sub macroblocks with sub macroblock type Pred_x_NxM or Pred_x_y_NxM or BiPred_x_NxM or BiPred_x_y_NxM, x, y = L0, L1, Bi, sub_block_type is NxM.

For macroblocks and sub macroblocks that are predicted in direct mode, sub_block_type is derived from the co-located macroblock or sub macroblock. If MbABTFlag is equal 1 for the co-located macroblock, sub_block_type for the current macroblock is derived from the co-located macroblock type or sub macroblock type as specified above. If MbABTFlag is equal 0 for the co-located macroblock and the type of the co-located macroblock is not Intra_16x16, sub_block_type is equal 4x4. If MbABTFlag is equal 0 for the co-located macroblock and the type of the co-located macroblock is Intra_16x16, sub_block_type is equal 8x8. If the co-located macroblock type is MbSkip, sub_block_type is equal 8x8.

block_coding_type indicates if an Inter or an Intra block is decoded. Inter blocks are indicated by block_coding_type = 'InterABT', intra blocks are indicated by block_coding_type = 'IntraABT'.

12.3.1.3.1 Residual sub block CAVLC semantics

~~num_coeffABT~~**num_coeff_abt** indicates the number of coefficients to be decoded. ~~num_coeffABT~~num_coeff_abt is bound by max_num_coeff_abt( ) specified in Table 12-3.

code_number indicates the number of the decoded codeword. The code structure of the codewords depending on the syntax element to be decoded is specified in subclause 12.4.1.

code_number2run_intra( ) retrieves run from Table 12-~~18~~9 for Intra blocks dependent on QP_Y.

code_number2run_inter( ) retrieves run from Table 12-~~18~~9 for Inter blocks.

code_number2level_intra( ) retrieves level from Table 12-~~18~~9 for Intra blocks dependent on QP_Y.

code_number2level_inter( ) retrieves level from Table 12-~~18~~9 for Inter blocks.

escape indicates separate( ) decoding of level and run. The value of escape is specified in subclause 12.5.1.2.2.

escape_level( ) retrieve level symbol after escape as specified in Table 12-~~17~~8.

escape_run( ) retrieve run symbol after escape as specified in Table 12-~~15~~6.

12.3.1.3.2 Residual sub block CABAC semantics

~~[Ed.Note: consult Detlev]~~ The syntax elements of residual_subblock_cabac( ) are specified in subclause 7.4.5.3.2.

12.4 ABT decoding process

12.4.1 Intra Prediction for 4x8, 8x4, and 8x8 luma blocks

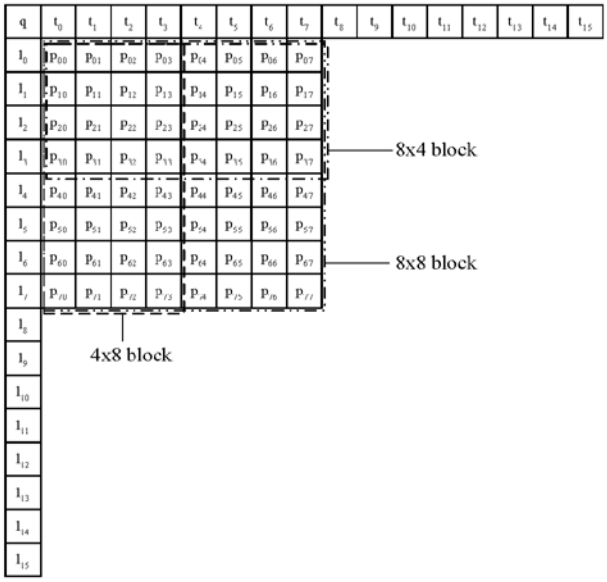

Figure 12-2 – Identification of samples used for ABT intra spatial prediction for 4x8, 8x4, and 8x8 luma blocks

Figure 12-2 illustrates the intra prediction for 4x8, 8x4, and 8x8 blocks that may be used in addition to the 4x4 intra prediction specified in subclause 8.5. The samples p_{mn}, with m=0 to M-1 and n=0 to N-1, M,N = {4,8}, are predicted using samples t_k, k=0 to (N+M-1), q, and l_k, k=0 to (N+M-1), from neighbouring blocks.

Samples t_k, k=0 to (N+M-1) or samples l_k, k=0 to (N+M-1) shall be considered not available under the following circumstances:

- if they are outside the picture or outside the current slice,
- if they belong to a macroblock that is subsequent to the current macroblock in raster scan order,
- if they are sent later than the current block in the order shown in Figure 12-1, or
- if they are in a non-intra macroblock and constrained_intra_pred is 1.

When samples t_k, k=N to (N+M-1) are not available the sample value of t_{N-1} is substituted for the samples t_k, k=N to (N+M-1). When samples l_k, k=M to (N+M-1) are not available the sample value of l_{M-1} is substituted for the samples l_k, k=N to (N+M-1).

12.4.1.1 Mode 0: vertical prediction

t_k, k = 0 to (N-1) shall be available. Prediction sample q is denoted as t_{-1}. If q is not in the slice, t_0 substitutes q.

block size		samples	predicted by
4x8	k = 0 to 7, i = 0 to 3	p_{ki}	$(t_{i-1} + t_i<<1 + t_{i+1} + 2) >> 2$
8x4	k = 0 to 3, i = 0 to 7		
8x8	k = 0 to 7, i = 0 to 7		

12.4.1.2 Mode 1: horizontal prediction

l_k, k = 0 to (M-1) shall be available. Prediction sample q is denoted as l_{-1}. If q is not in the slice, l_0 substitutes q.

block size		samples	predicted by
4x8	k = 0 to 7, i = 0 to 3	p_{ki}	$(l_{i-1} + l_i<<1 + l_i+1 + 2) >> 2$
8x4	k = 0 to 3, i = 0 to 7		
8x8	k = 0 to 7, i = 0 to 7		

12.4.1.3 Mode 2: DC prediction

For DC prediction, all samples p_{ki} ~~in P~~ are predicted by the same value p.

a) If l_k, k = 0 to (M-1) are available and t_k, k = 0 to (N-1) are available

block size	p
4x8	$((t_0 - t_1 + t_2 + t_3 + 2) >> 2 + (l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + 4) >> 3) >> 1$
8x4	$((t_0 - t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + 4) >> 3 + (l_0 + l_1 + l_2 + l_3 + 2) >> 2) >> 1$
8x8	$(t_0 + t_1 + t_2 + t_3 + l_4 + t_5 + t_6 + t_7 + l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + 8) >> 4$

b) If l_k, k = 0 to (M-1) are available and t_k, k = 0 to (N-1) are not available

block size	p
4x8	$(l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + 4) >> 3$
8x4	$(l_0 + l_1 + l_2 + l_3 + 2) >> 2$
8x8	$(l_0 + l_1 + l_2 + l_3 + l_4 + l_5 + l_6 + l_7 + 4) >> 3$

c) If l_k, k = 0 to (M-1) are not available and t_k, k = 0 to (N-1) are available

block size	p
4x8	$(t_0 + t_1 + t_2 + t_3 + 2) >> 2$
8x4	$(t_0 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + 4) >> 3$

8x8	$(t_0 + t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + 4) >> 3$

d) If l_k, k = 0 to (M-1) are not available and t_k, k = 0 to (N-1) are not available

block size	p
4x8	128
8x4	128
8x8	128

12.4.1.4 Mode 3: diagonal down/left prediction

t_k, k = 0 to (N-1) shall be available, and l_k, k = 0 to (M-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
p_{03}	p_{00}	p_{00}	$((l_2 + l_1<<1 + l_0 + 2) >> 2 + (t_0 + t_1<<1 + t_2 + 2) >> 2) >> 1$
p_{01}, p_{10}	p_{01}, p_{10}	p_{01}, p_{10}	$((l_3 + l_2<<1 + l_1 + 2) >> 2 + (t_1 + t_2<<1 + t_3 + 2) >> 2) >> 1$
p_{02}, p_{11}, p_{20}	p_{02}, p_{11}, p_{20}	p_{02}, p_{11}, p_{20}	$((l_4 + l_3<<1 + l_2 + 2) >> 2 + (t_2 + t_3<<1 + t_4 + 2) >> 2) >> 1$
p_{03}, p_{12}, p_{21}, p_{30}	p_{03}, p_{12}, p_{21}, p_{30}	p_{03}, p_{12}, p_{21}, p_{30}	$((l_5 + l_4<<1 + l_3 + 2) >> 2 + (t_3 + t_4<<1 + t_5 + 2) >> 2) >> 1$
p_{13}, p_{22}, p_{31}, p_{40}	p_{04}, p_{13}, p_{22}, p_{31}	p_{04}, p_{13}, p_{22}, p_{31}, p_{40}	$((l_6 + l_5<<1 + l_4 + 2) >> 2 + (t_4 + t_5<<1 + t_6 + 2) >> 2) >> 1$
p_{23}, p_{32}, p_{41}, p_{50}	p_{05}, p_{14}, p_{23}, p_{32}	p_{05}, p_{14}, p_{23}, p_{32}, p_{41}, p_{50}	$((l_7 + l_6<<1 + l_5 + 2) >> 2 + (t_5 + t_6<<1 + t_7 + 2) >> 2) >> 1$
p_{33}, p_{42}, p_{51}, p_{60}	p_{06}, p_{15}, p_{24}, p_{33}	p_{06}, p_{15}, p_{24}, p_{33}, p_{42}, p_{51}, p_{60}	$((l_8 + l_7<<1 + l_6 + 2) >> 2 + (t_6 + t_7<<1 + t_8 + 2) >> 2) >> 1$
p_{43}, p_{52}, p_{61}, p_{70}	p_{07}, p_{16}, p_{25}, p_{34}	p_{07}, p_{16}, p_{25}, p_{34}, p_{43}, p_{52}, p_{61}, p_{70}	$((l_9 + l_8<<1 + l_7 + 2) >> 2 + (t_7 + t_8<<1 + t_9 + 2) >> 2) >> 1$
p_{53}, p_{62}, p_{71}	p_{17}, p_{26}, p_{35}	p_{17}, p_{26}, p_{35}, , p_{44}, p_{53}, p_{62}, p_{71}	$((l_{10} + l_9<<1 + l_8 + 2) >> 2 + (t_8 + t_9<<1 + t_{10} + 2) >> 2) >> 1$
p_{63}, p_{72}	p_{27}, p_{36}	p_{27}, p_{36}, p_{45}, p_{54}, p_{63}, p_{72}	$((l_{11} + l_{10}<<1 + l_9 + 2) >> 2 + (t_9 + t_{10}<<1 + t_{11} + 2) >> 2) >> 1$
p_{73}	p_{37}	p_{37}, p_{46}, p_{55}, p_{64}, p_{73}	$((l_{12} + l_{11}<<1 + l_{10} + 2) >> 2 + (t_{10} - t_{11}<<1 + t_{12} + 2) >> 2) >> 1$
-	-	p_{47}, p_{56}, p_{65}, p_{74}	$((l_{13} + l_{12}<<1 + l_{11} + 2) >> 2 + (t_{11} - t_{12}<<1 + t_{13} + 2) >> 2) >> 1$
-	-	p_{57}, p_{66}, p_{75}	$((l_{14} + l_{13}<<1 + l_{12} + 2) >> 2 + (t_{12} - t_{13}<<1 + t_{14} + 2) >> 2) >> 1$
-	-	p_{67}, p_{76}	$((l_{15} + l_{14}<<1 + l_{13} + 2) >> 2 + (t_{13} - t_{14}<<1 + t_{15} + 2) >> 2) >> 1$
-	-	p_{77}	$((l_{15} + l_{15}<<1 + l_{14} + 2) >> 2 + (t_{14} - t_{15}<<1 + t_{15} + 2) >> 2) >> 1$

12.4.1.5 Mode 4: diagonal down/right prediction

t_k, k = 0 to (N-1) shall be available, and q, and l_k, k = 0 to (M-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
-	p_{07}	p_{07}	$(t_5 + t_6<<1 + t_7 + 2) >> 2$
-	p_{06}, p_{17}	p_{06}, p_{17}	$(t_4 + t_5<<1 + t_6 + 2) >> 2$
-	p_{05}, p_{16}, p_{27}	p_{05}, p_{16}, p_{27}	$(t_3 + t_4<<1 + t_5 + 2) >> 2$
-	p_{04}, p_{15}, p_{26}, p_{37}	p_{04}, p_{15}, p_{26}, p_{37}	$(t_2 + t_3<<1 + t_4 + 2) >> 2$
p_{03}	p_{03}, p_{14}, p_{25}, p_{36}	p_{03}, p_{14}, p_{25}, p_{36}, p_{47}	$(t_1 + t_2<<1 + t_3 + 2) >> 2$
p_{02}, p_{13}	p_{02}, p_{13}, p_{24}, p_{35}	p_{02}, p_{13}, p_{24}, p_{35}, p_{46}, p_{57}	$(t_0 + t_1<<1 + t_2 + 2) >> 2$
p_{01}, p_{12}, p_{23}	p_{01}, p_{12}, p_{23}, p_{34}	p_{01}, p_{12}, p_{23}, p_{34}, , p_{45}, p_{56}, p_{67}	$(q + t_0<<1 + t_1 + 2) >> 2$
p_{00}, p_{11}, p_{22}, p_{33}	p_{00}, p_{11}, p_{22}, p_{33}	p_{00}, p_{11}, p_{22}, p_{33}, p_{44}, p_{55}, p_{66}, p_{77}	$(l_0 + q<<1 + t_0 + 2) >> 2$

p_{10}, p_{21}, p_{32}, p_{43}	p_{10}, p_{21}, p_{32}	p_{10}, p_{21}, p_{32}, , p_{43}, p_{54}, p_{65}, p_{76}	$(l_1 + l_0 << 1 + q + 2) >> 2$
p_{20}, p_{31}, p_{42}, p_{53}	p_{20}, p_{31}	p_{20}, p_{31}, p_{42}, p_{53}, p_{64}, p_{75}	$(l_2 + l_1 << 1 + l_0 + 2) >> 2$
p_{30}, p_{41}, p_{52}, p_{63}	p_{30}	p_{30}, p_{41}, p_{52}, p_{53}, p_{74}	$(l_3 + l_2 << 1 + l_1 + 2) >> 2$
p_{40}, p_{51}, p_{62}, p_{73}	-	p_{40}, p_{51}, p_{62}, p_{73}	$(l_4 + l_3 << 1 + l_2 + 2) >> 2$
p_{50}, p_{61}, p_{72}	-	p_{50}, p_{61}, p_{72}	$(l_5 + l_4 << 1 + l_3 + 2) >> 2$
p_{60}, p_{71}	-	p_{60}, p_{71}	$(l_6 + l_5 << 1 + l_4 + 2) >> 2$
p_{70}	-	p_{70}	$(l_7 + l_6 << 1 + l_5 + 2) >> 2$

12.4.1.6 Mode 5: vertical-left prediction

t_k, k = 0 to (N-1) shall be available, and q, and l_k, k = 0 to (M-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
-	p_{07}	p_{07}	$((t_6 + t_7 << 1 + t_8 + 2) >> 2 + (t_7 + t_8 << 1 + t_9 + 2) >> 2) >> 1$
-	p_{17}	p_{17}	$(t_6 + t_7 << 1 + t_8 + 2) >> 2$
-	p_{06}, p_{27}	p_{06}, p_{27}	$((t_5 + t_6 << 1 + t_7 + 2) >> 2 + (t_6 + t_7 << 1 + t_8 + 2) >> 2) >> 1$
-	p_{16}, p_{37}	p_{16}, p_{37}	$(t_5 + t_6 << 1 + t_7 + 2) >> 2$
-	p_{05}, p_{26}	p_{05}, p_{26}, p_{47}	$((t_4 + t_5 << 1 + t_6 + 2) >> 2 + (t_5 + t_6 << 1 + t_7 + 2) >> 2) >> 1$
-	p_{15}, p_{36}	p_{15}, p_{36}, p_{57}	$(t_4 + t_5 << 1 + t_6 + 2) >> 2$
-	p_{04}, p_{25}	p_{04}, p_{25}, p_{46}, p_{67}	$((t_3 + t_4 << 1 + t_5 + 2) >> 2 + (t_4 + t_5 << 1 + t_6 + 2) >> 2) >> 1$
-	p_{14}, p_{35}	p_{14}, p_{35}, p_{56}, p_{77}	$(t_3 + t_4 << 1 + t_5 + 2) >> 2$
p_{03}	p_{03}, p_{24}	p_{03}, p_{24}, p_{45}, p_{66}	$((t_2 + t_3 << 1 + t_4 + 2) >> 2 + (t_3 + t_4 << 1 + t_5 + 2) >> 2) >> 1$
p_{13}	p_{13}, p_{34}	p_{13}, p_{34}, p_{55}, p_{76}	$(t_2 + t_3 << 1 + t_4 + 2) >> 2$
p_{02}, p_{23}	p_{02}, p_{23}	p_{02}, p_{23}, p_{44}, p_{65}	$((t_1 + t_2 << 1 + t_3 + 2) >> 2 + (t_2 + t_3 << 1 + t_4 + 2) >> 2) >> 1$
p_{12}, p_{33}	p_{12}, p_{33}	p_{12}, p_{33}, p_{54}, p_{75}	$(t_1 + t_2 << 1 + t_3 + 2) >> 2$
p_{01}, p_{22}, p_{43}	p_{01}, p_{22}	p_{01}, p_{22}, p_{43}, p_{64}	$((t_0 + t_1 << 1 + t_2 + 2) >> 2 + (t_1 + t_2 << 1 + t_3 + 2) >> 2) >> 1$
p_{11}, p_{32}, p_{53}	p_{11}, p_{32}	p_{11}, p_{32}, p_{53}, p_{74}	$(t_0 + t_1 << 1 + t_2 + 2) >> 2$
p_{00}, p_{21}, p_{42}, p_{63}	p_{00}, p_{21}	p_{00}, p_{21}, p_{42}, p_{63}	$((l_0 + q << 1 + t_0 + 2) >> 2 - (t_0 + t_1 << 1 + t_2 + 2) >> 2) >> 1$
p_{10}, p_{31}, p_{52}, p_{73}	p_{10}, p_{31}	p_{10}, p_{31}, p_{52}, p_{73}	$(l_0 + q << 1 - t_0 + 2) >> 2$
p_{20}, p_{41}, p_{62}	p_{20}	p_{20}, p_{41}, p_{62}	$(l_2 + l_1 << 1 + l_0 + 2) >> 2$
p_{30}, p_{51}, p_{72}	p_{30}	p_{30}, p_{51}, p_{72}	$(l_3 + l_2 << 1 + l_1 + 2) >> 2$
p_{40}, p_{61}	-	p_{40}, p_{61}	$(l_4 + l_3 << 1 + l_2 + 2) >> 2$
p_{50}, p_{71}	-	p_{50}, p_{71}	$(l_5 + l_4 << 1 + l_3 + 2) >> 2$
p_{60}	-	p_{60}	$(l_6 + l_5 << 1 + l_4 + 2) >> 2$
p_{70}	-	p_{70}	$(l_7 + l_6 << 1 + l_5 + 2) >> 2$

12.4.1.7 Mode 6: horizontal-down prediction

t_k, k = 0 to (N-1) shall be available, and q, and l_k, k = 0 to (M-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
-	p_{07}	p_{07}	$(t_5 + t_6 << 1 + t_7 + 2) >> 2$

-	p_{06}	p_{06}	$(t_4 + t_5 << 1 + t_6 + 2) >> 2$
-	p_{05}, p_{17}	p_{05}, p_{17}	$(t_3 + t_4 << 1 + t_5 + 2) >> 2$
-	p_{04}, p_{16}	p_{04}, p_{16}	$(t_2 + t_3 << 1 + t_4 + 2) >> 2$
p_{03}	p_{03}, p_{15}, p_{27}	p_{03}, p_{15}, p_{27}	$(t_1 + t_2 << 1 + t_3 + 2) >> 2$
p_{02}	p_{02}, p_{14}, p_{26}	p_{02}, p_{14}, p_{26}	$(t_0 + t_1 << 1 + t_2 + 2) >> 2$
p_{01}, p_{13}	p_{01}, p_{13}, p_{25}, p_{37}	p_{01}, p_{13}, p_{25}, p_{37}	$(q + t_0 << 1 + t_1 + 2) >> 2$
p_{00}, p_{12}	p_{00}, p_{12}, p_{24}, p_{36}	p_{00}, p_{12}, p_{24}, p_{36}	$((l_0 + q << 1 + t_0 + 2) >> 2 - (q + l_0 << 1 + l_1 + 2) >> 2) >> 1$
p_{10}, p_{22}	p_{10}, p_{22}, p_{34}	p_{10}, p_{22}, p_{34}, p_{46}	$((q - l_0 << 1 + l_1 + 2) >> 2 - (l_0 + l_1 << 1 + l_2 + 2) >> 2) >> 1$
p_{11}, p_{23}	p_{11}, p_{23}, p_{35}	p_{11}, p_{23}, p_{35}, p_{47}	$(q + l_0 << 1 + l_1 + 2) >> 2$
p_{20}, p_{32}	p_{20}, p_{32}	p_{20}, p_{32}, p_{44}, p_{56}	$((l_0 + l_1 << 1 + l_2 + 2) >> 2 + (l_1 + l_2 << 1 + l_3 + 2) >> 2) >> 1$
p_{21}, p_{33}	p_{21}, p_{33}	p_{21}, p_{33}, p_{45}, p_{57}	$(l_0 + l_1 << 1 + l_2 + 2) >> 2$
p_{30}, p_{42}	p_{30}	p_{30}, p_{42}, p_{54}, p_{66}	$((l_1 + l_2 << 1 + l_3 + 2) >> 2 + (l_2 + l_3 << 1 + l_4 + 2) >> 2) >> 1$
p_{31}, p_{43}	p_{31}	p_{31}, p_{43}, p_{55}, p_{67}	$(l_1 + l_2 << 1 + l_3 + 2) >> 2$
p_{40}, p_{52}	-	p_{46}, p_{52}, p_{64}, p_{76}	$((l_2 + l_3 << 1 + l_4 + 2) >> 2 + (l_3 + l_4 << 1 + l_5 + 2) >> 2) >> 1$
p_{41}, p_{53}	-	p_{41}, p_{53}, p_{65}, p_{77}	$(l_2 + l_3 << 1 + l_4 + 2) >> 2$
p_{50}, p_{62}	-	p_{50}, p_{62}, p_{74}	$((l_3 + l_4 << 1 + l_5 + 2) >> 2 + (l_4 + l_5 << 1 + l_6 + 2) >> 2) >> 1$
p_{51}, p_{63}	-	p_{51}, p_{63}, p_{75}	$(l_3 + l_4 << 1 + l_5 + 2) >> 2$
p_{60}, p_{72}	-	p_{60}, p_{72}	$((l_4 + l_5 << 1 + l_6 + 2) >> 2 + (l_5 + l_6 << 1 + l_7 + 2) >> 2) >> 1$
p_{61}, p_{73}	-	p_{61}, p_{73}	$(l_4 + l_5 << 1 + l_6 + 2) >> 2$
p_{70}	-	p_{76}	$((l_5 + l_6 << 1 + l_7 + 2) >> 2 + (l_6 + l_7 << 1 + l_8 + 2) >> 2) >> 1$
p_{71}	-	p_{71}	$(l_5 + l_6 << 1 + l_7 + 2) >> 2$

12.4.1.8 Mode 7: vertical-right prediction

t_k, k = 0 to (N-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
p_{00}	p_{00}	p_{00}	$((t_0 + t_1 << 1 + t_2 - 2) >> 2 + (t_1 + t_2 << 1 + t_3 + 2) >> 2) >> 1$
p_{10}	p_{10}	p_{10}	$(t_1 + t_2 << 1 + t_3 + 2) >> 2$
p_{01}, p_{20}	p_{01}, p_{20}	p_{01}, p_{20}	$((t_1 + t_2 << 1 + t_3 - 2) >> 2 + (t_2 + t_3 << 1 + t_4 + 2) >> 2) >> 1$
p_{11}, p_{30}	p_{11}, p_{30}	p_{11}, p_{30}	$(t_2 + t_3 << 1 + t_4 + 2) >> 2$
p_{02}, p_{21}, p_{40}	p_{02}, p_{21}	p_{02}, p_{21}, p_{40}	$((t_2 + t_3 << 1 + t_4 - 2) >> 2 + (t_3 + t_4 << 1 + t_5 + 2) >> 2) >> 1$
p_{12}, p_{31}, p_{50}	p_{12}, p_{31}	p_{12}, p_{31}, p_{50}	$(t_3 + t_4 << 1 + t_5 + 2) >> 2$
p_{03}, p_{22}, p_{41}, p_{60}	p_{03}, p_{22}	p_{03}, p_{22}, p_{41}, p_{60}	$((t_3 + t_4 << 1 + t_5 - 2) >> 2 + (t_4 + t_5 << 1 + t_6 + 2) >> 2) >> 1$
p_{13}, p_{32}, p_{51}, p_{70}	p_{13}, p_{32}	p_{13}, p_{32}, p_{51}, p_{70}	$(t_4 + t_5 << 1 + t_6 + 2) >> 2$
p_{23}, p_{42}, p_{61}	p_{04}, p_{23}	p_{04}, p_{23}, p_{42}, p_{61}	$((t_0 + t_1 << 1 + t_2 - 2) >> 2 + (t_1 + t_2 << 1 + t_3 + 2) >> 2) >> 1$
p_{33}, p_{52}, p_{71}	p_{14}, p_{33}	p_{14}, p_{33}, p_{52}, p_{71}	$(t_5 + t_6 << 1 + t_7 + 2) >> 2$
p_{43}, p_{62}	p_{05}, p_{24}	p_{05}, p_{24}, p_{43}, p_{62}	$((t_5 + t_5 << 1 + t_7 - 2) >> 2 + (t_6 + t_7 << 1 + t_8 + 2) >> 2) >> 1$
p_{53}, p_{72}	p_{15}, p_{34}	p_{15}, p_{34}, p_{53}, p_{72}	$(t_6 + t_7 << 1 + t_8 + 2) >> 2$

p_{63}	p_{06}, p_{25}	p_{06}, p_{25}, p_{44}, p_{63}	$((t_6 + t_7 << 1 + t_8 - 2) >> 2 + (t_7 + t_8 << 1 + t_9 + 2) >> 2) >> 1$
p_{73}	p_{16}, p_{35}	p_{16}, p_{35}, p_{54}, p_{73}	$(t_7 + t_8 << 1 + t_9 + 2) >> 2$
-	p_{07}, p_{26}	p_{07}, p_{26}, p_{45}, p_{64}	$((t_7 + t_8 << 1 + t_9 - 2) >> 2 + (t_8 + t_9 << 1 + t_{10} + 2) >> 2) >> 1$
-	p_{17}, p_{36}	p_{17}, p_{36}, p_{55}, p_{74}	$(t_8 + t_9 << 1 + t_{10} + 2) >> 2$
-	p_{27}	p_{27}, p_{46}, p_{65}	$((t_8 + t_9 << 1 + t_{10} + 2) >> 2 + (t_9 + t_{10} << 1 + t_{11} + 2) >> 2) >> 1$
-	p_{37}	p_{37}, p_{56}, p_{75}	$(t_9 + t_{10} << 1 + t_{11} + 2) >> 2$
-	-	p_{47}, p_{66}	$((t_9 + t_{10} << 1 + t_{11} + 2) >> 2 + (t_{10} + t_{11} << 1 + t_{12} - 2) >> 2) >> 1$
-	-	p_{57}, p_{76}	$(t_{10} + t_{11} << 1 + t_{12} + 2) >> 2$
-	-	p_{67}	$((t_{10} + t_{11} << 1 + t_{12} + 2) >> 2 + (t_{11} + t_{12} << 1 + t_{13} + 2) >> 2) >> 1$
-	-	p_{77}	$(t_{11} + t_{12} << 1 + t_{13} + 2) >> 2$

12.4.1.9 Mode 8: horizontal-up prediction

l_k, k = 0 to (M-1) shall be available.

4x8 block samples	8x4 block samples	8x8 block samples	predicted by
p_{00}	p_{00}	p_{00}	$((l_0 + l_1 << 1 + l_2 - 2) >> 2 + (l_1 + l_2 << 1 + l_3 + 2) >> 2) >> 1$
p_{01}	p_{01}	p_{01}	$(l_1 + l_2 << 1 + l_3 + 2) >> 2$
p_{10}, p_{02}	p_{10}, p_{32}	p_{10}, p_{02}	$((l_1 + l_2 << 1 + l_3 - 2) >> 2 + (l_2 + l_3 << 1 + l_4 + 2) >> 2) >> 1$
p_{11}, p_{03}	p_{11}, p_{33}	p_{11}, p_{03}	$(l_2 + l_3 << 1 + l_4 + 2) >> 2$
p_{20}, p_{12}	p_{20}, p_{12}, p_{04}	p_{20}, p_{12}, p_{04}	$((l_2 + l_3 << 1 + l_4 - 2) >> 2 + (l_3 + l_4 << 1 + l_5 + 2) >> 2) >> 1$
p_{21}, p_{13}	p_{21}, p_{13}, p_{05}	p_{21}, p_{13}, p_{05}	$(l_3 + l_4 << 1 + l_5 + 2) >> 2$
p_{30}, p_{22}	p_{30}, p_{22}, p_{14}, p_{06}	p_{30}, p_{22}, p_{14}, p_{06}	$((l_3 + l_4 << 1 + l_5 - 2) >> 2 + (l_4 + l_5 << 1 + l_6 + 2) >> 2) >> 1$
p_{31}, p_{23}	p_{31}, p_{23}, p_{15}, p_{07}	p_{31}, p_{23}, p_{15}, p_{07}	$(l_4 + l_5 << 1 + l_6 + 2) >> 2$
p_{40}, p_{32}	p_{32}, p_{24}, p_{16}	p_{40}, p_{32}, p_{24}, p_{16}	$((l_4 + l_5 << 1 + l_6 - 2) >> 2 + (l_5 + l_6 << 1 + l_7 + 2) >> 2) >> 1$
p_{41}, p_{33}	p_{33}, p_{25}, p_{17}	p_{41}, p_{33}, p_{25}, p_{17}	$(l_5 + l_6 << 1 + l_7 + 2) >> 2$
p_{50}, p_{42}	p_{34}, p_{26}	p_{50}, p_{42}, p_{34}, p_{26}	$((l_5 + l_6 << 1 + l_7 - 2) >> 2 + (l_6 + l_7 << 1 + l_8 + 2) >> 2) >> 1$
p_{51}, p_{43}	p_{35}, p_{27}	p_{51}, p_{43}, p_{35}, p_{27}	$(l_6 + l_7 << 1 + l_8 + 2) >> 2$
p_{60}, p_{52}	p_{36}	p_{60}, p_{52}, p_{44}, p_{36}	$((l_6 + l_7 << 1 + l_8 - 2) >> 2 + (l_7 + l_8 << 1 + l_9 + 2) >> 2) >> 1$
p_{61}, p_{53}	p_{37}	p_{61}, p_{53}, p_{45}, p_{37}	$(l_7 + l_8 << 1 + l_9 + 2) >> 2$
p_{70}, p_{62}	-	p_{70}, p_{62}, p_{54}, p_{46}	$((l_7 + l_8 << 1 + l_9 - 2) >> 2 + (l_8 + l_9 << 1 + l_{10} + 2) >> 2) >> 1$
p_{71}, p_{63}	-	p_{71}, p_{63}, p_{55}, p_{47}	$(l_8 + l_9 << 1 + l_{10} + 2) >> 2$
p_{72}	-	p_{72}, p_{64}, p_{56}	$((l_8 + l_9 << 1 + l_{10} + 2) >> 2 + (l_9 + l_{10} << 1 + l_{11} + 2) >> 2) >> 1$
p_{73}	-	p_{73}, p_{65}, p_{57}	$(l_9 + l_{10} << 1 + l_{11} + 2) >> 2$
-	-	p_{74}, p_6	$((l_9 + l_{10} << 1 + l_{11} + 2) >> 2 + (l_{10} + l_{11} << 1 + l_{12} - 2) >> 2) >> 1$
-	-	p_{75}, p_{67}	$(l_{10} + l_{11} << 1 + l_{12} + 2) >> 2$
-	-	p_{76}	$((l_{10} + l_{11} << 1 + l_{12} + 2) >> 2 + (l_{11} + l_{12} << 1 + l_{13} + 2) >> 2) >> 1$
-	-	p_{77}	$(l_{11} + l_{12} << 1 + l_{13} + 2) >> 2$

12.4.2 Scanning method for ABT blocks

Scanning patterns for blocks of size 4x4, 4x8, 8x4, and 8x8 coefficients are given below. For blocks decoded in frame mode, the zig-zag scans are used. For block decoded in field mode, the field scans are used. The zig-zag scan for 4x4 blocks corresponds to the zig-zag scan specified in Figure 8-12.

12.4.2.1 Zig-zag scan

0	1	5	6
2	4	7	12
3	8	11	13
9	10	14	15

Figure 12-3 – 4x4 zig-zag scan

0	2	3	9
1	4	8	10
5	7	11	17
6	12	16	18
13	15	19	25
14	20	24	26
21	23	27	30
22	28	29	31

Figure 12-4 – 4x8 zig-zag scan

0	1	5	6	13	14	21	22
2	4	7	12	15	20	23	28
3	8	11	16	19	24	27	29
9	10	17	18	25	26	30	31

Figure 12-5 – 8x4 zig-zag scan

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Figure 12-6 – 8x8 zig-zag scan

12.4.2.2 Field scan

0	2	8	12
1	5	9	13
3	6	10	14
4	7	11	15

Figure 12-7 – 4x4 field scan

0	4	12	20
1	5	13	21
2	6	14	22
3	11	19	27
7	15	23	28
8	16	24	29
9	17	25	30
10	18	26	31

Figure 12-8 – 4x8 field scan

0	2	6	10	14	18	22	26
1	5	9	13	17	21	25	29
3	7	11	15	19	23	27	30
4	8	12	16	20	24	28	31

Figure 12-9 – 8x4 field scan

0	3	8	15	22	30	38	52
1	4	14	21	29	37	45	53
2	7	16	23	31	39	46	58
5	9	20	28	36	44	51	59
6	13	24	32	40	47	54	60
10	17	25	33	41	48	55	61
11	18	26	34	42	49	56	62
12	19	27	35	43	50	57	63

Figure 12-10 – 8x8 field scan

12.4.3 Scaling and inverse transform for ABT blocks

The scaling and inverse transform of residual blocks of block size larger than 4x4 is specified below. The scaling and inverse transform for 4x4 blocks is specified in subclause 8.6. For 8x8 blocks, the coefficients $R_{ij}^{(m)}$, used in the formulas below, are defined as:

$$R_{ij}^{(m)} = V_m^{8x8} \tag{12-3}$$

where the subscript of V^{8x8} is the row index of the vector defined as:

$$V^{8x8} = \begin{bmatrix} 15 \\ 17 \\ 19 \\ 22 \\ 24 \\ 27 \end{bmatrix}. \tag{12-4}$$

For 4x8 blocks, the coefficients $R_{ij}^{(m)}$, used in the formulas below, are defined as:

$$R_{ij}^{(m)} = \begin{cases} V_{m0}^{8x4,4x8} & \text{for} \quad i = 0,\ldots,7;\ j = 0,2 \\ V_{m1}^{8x4,4x8} & \text{for} \quad i = 0,\ldots,7;\ j = 1,3 \end{cases} \tag{12-5}$$

where the first and second subscripts of $V^{8x4,4x8}$ are row and column indices, respectively, of the matrix defined as:

$$V^{8x4,4x8} = \begin{bmatrix} 9 & 11 \\ 10 & 12 \\ 11 & 14 \\ 12 & 16 \\ 14 & 17 \\ 15 & 20 \end{bmatrix}. \tag{12-6}$$

For 8x4 blocks, the coefficients $R_{ij}^{(m)}$, used in the formulas below, are defined as:

$$R_{ij}^{(m)} = \begin{cases} V_{m0}^{8x4,4x8} & \text{for} \quad i = 0, 2;\ j = 0,...,7 \\ V_{m1}^{8x4,4x8} & \text{for} \quad i = 1, 3;\ j = 0,...,7 \end{cases} \tag{12-7}$$

where the first and second subscripts of $V^{8x4,4x8}$ are row and column indices, respectively, of the matrix defined in Equation 12-6.

The coefficient levels are multiplied with the scaling value R

$$w_{ij} = [c_{ij} \cdot R_{ij}^{(QP\%6)}] << (QP/6 - 2), \quad i = 0,...,N, j = 0,...,M \tag{12-8}$$

After constructing an entire MxN block of scaled transform coefficients and assembling these into a MxN matrix W of elements w_{ij} illustrated as

$$W = \begin{bmatrix} w_{00} & \cdots & w_{0(N-1)} \\ \vdots & \ddots & \vdots \\ w_{(M-1)0} & \cdots & w_{(M-1)(N-1)} \end{bmatrix} \tag{12-9}$$

W is inverse transformed horizontally. If N = = 4, the one-dimensional inverse transform is performed as specified in subclause 8.6.2.3. If N = = 8, the inverse transform is specified by Equation 12-10,

$$Z' = \begin{bmatrix} w_{00} & \cdots & w_{07} \\ \vdots & \ddots & \vdots \\ w_{(M-1)0} & \cdots & w_{(M-1)7} \end{bmatrix} \begin{bmatrix} 13 & 13 & 13 & 13 & 13 & 13 & 13 & 13 \\ 19 & 15 & 9 & 3 & -3 & -9 & -15 & -19 \\ 17 & 7 & -7 & -17 & -17 & -7 & 7 & 17 \\ 9 & 3 & -19 & -15 & 15 & 19 & -3 & -9 \\ 13 & -13 & -13 & 13 & 13 & -13 & -13 & 13 \\ 15 & -19 & -3 & 9 & -9 & 3 & 19 & -15 \\ 7 & -17 & 17 & -7 & -7 & 17 & -17 & 7 \\ 3 & -9 & 15 & -19 & 19 & -15 & 9 & -3 \end{bmatrix} \tag{12-10}$$

The result is rounded

$$Z_{ij} = \text{sign}(Z'_{ij})[\text{abs}(Z'_{ij}) + 2^{B_{\text{shift}}-1}] >> B_{\text{shift}} \tag{12-11}$$

where B_{shift} = = 7 for 8x8 blocks, and B_{shift} = = 2 for 4x8 or 8x4 blocks. If N = = 4, the second one-dimensional inverse transform is performed as specified in subclause 8.6.2.3. If N − − 8, the second inverse transform is specified by Equation 12-12 below,

$$X' = \begin{bmatrix} 13 & 19 & 17 & 9 & 13 & 15 & 7 & 3 \\ 13 & 15 & 7 & 3 & -13 & -19 & -17 & -9 \\ 13 & 9 & -7 & -19 & -13 & -3 & 17 & 15 \\ 13 & 3 & -17 & -15 & 13 & 9 & -7 & -19 \\ 13 & -3 & -17 & 15 & 13 & -9 & -7 & 19 \\ 13 & -9 & -7 & 19 & -13 & 3 & 17 & -15 \\ 13 & -15 & 7 & -3 & -13 & 19 & -17 & 9 \\ 13 & -19 & 17 & -9 & 13 & -15 & 7 & -3 \end{bmatrix} \begin{bmatrix} z_{00} & \cdots & z_{0(N-1)} \\ \vdots & \ddots & \vdots \\ z_{70} & \cdots & z_{7(N\ 1)} \end{bmatrix} \tag{12-12}$$

After the second (vertical) transform in Equation 12-12 step the final reconstructed sample residual values X'' shall be obtained as specified in Equation 8-59.